INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.1228

Fascicle 2/5

(09/97)

SERIES Q: SWITCHING AND SIGNALLING

Intelligent Network

Interface Recommendation for intelligent network Capability Set 2: Part 2

ITU-T Recommendation Q.1228 – Fascicle 2/5

(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 2

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

Recomendation Q.1228

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CAPABILITY SET 2

(Geneva, 1997)

PART 2

11 SSF application entity procedures

11.1 General

This clause provides the definition of the SSF Application Entity (AE) procedures related to the SSP-SCP interface. The procedures are based on the use of SS7; other signalling systems can be used (e.g. DSS 1 – layer 3).

Capabilities not explicitly covered by these procedures may be supported in an implementation dependent manner in the SSP, while remaining in line with clause 2.

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes TCAP (Transaction Capabilities Application Part) and one or more Application Service Elements (ASEs) called TC-users. The following subclauses define the TC-user ASE which interfaces with TCAP using the primitives specified in Recommendation Q.771; other signalling systems, such as DSS 1 (layer 3), may be used.

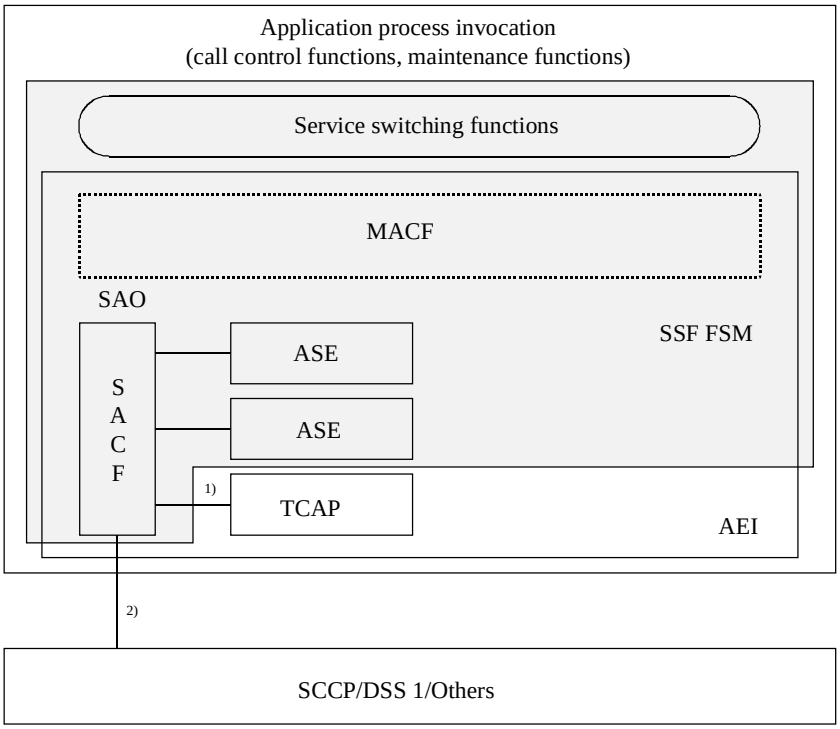
The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed clauses 17 and 18 shall be followed.

11.2 Model and interfaces

The functional model of the AE-SSF is shown in Figure 11-1; the ASEs interface to TCAP to communicate with the SCF, and interface to the Call Control Function (CCF) and the maintenance functions already defined for switching systems. The scope of this Recommendation is limited to the shaded area in Figure 11-1.

The interfaces shown in Figure 11-1 use the TC-user ASE primitives specified in Recommendation Q.771 and N-Primitives specified in Recommendation Q.711. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clauses 4 to 10.



AEI	Application Entity Invocation
SSF	Service Switching Functions
FSM	Finite State Machine
MACF	Multiple Assocation Control Function
SACF	Single Association Control Function
SAO	Single Association Object

[1] TC-Primitives.

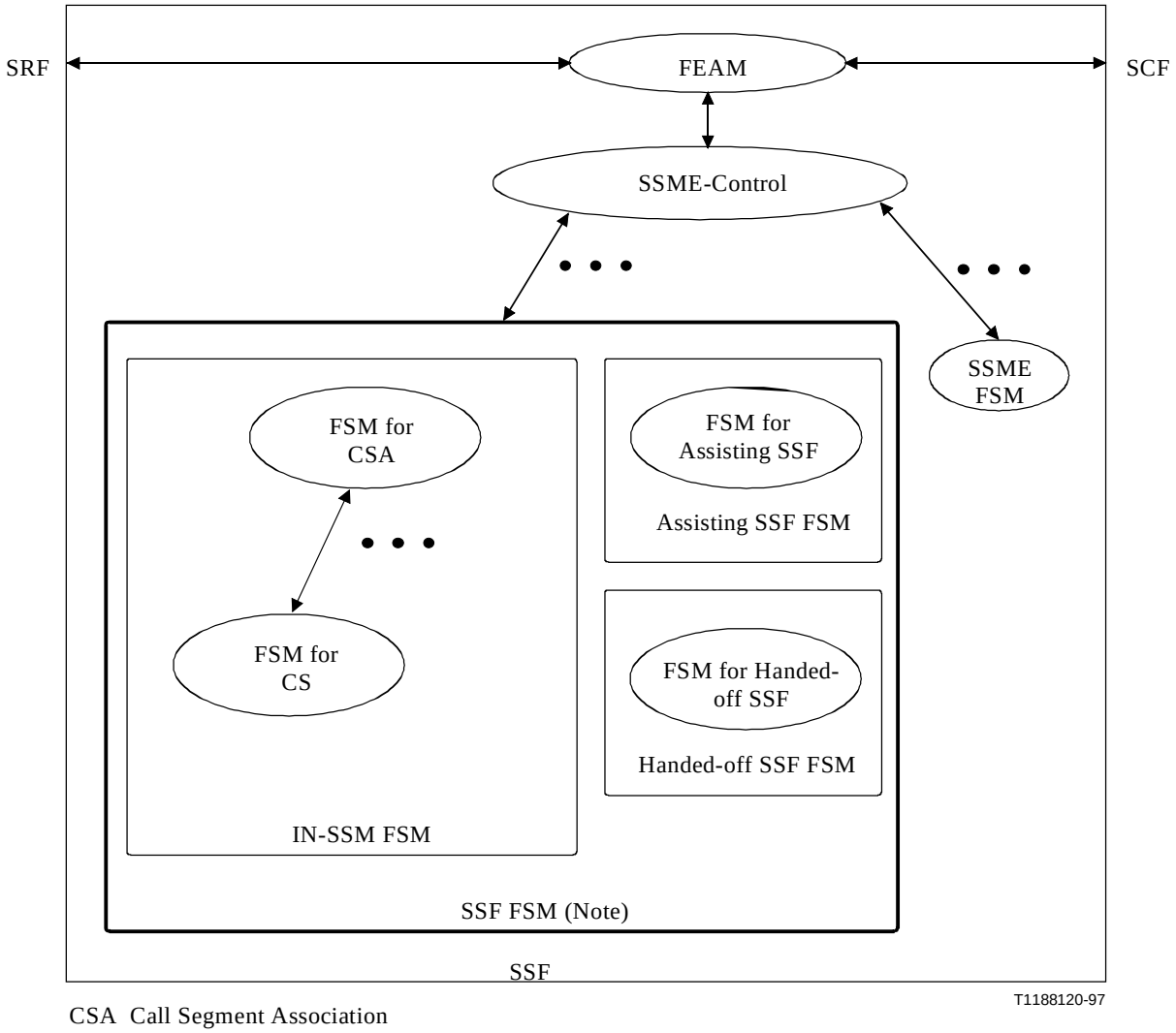
[2] N-Primitives.

NOTE – SSF FSM includes finite state machines.

Figure 11-1/Q.1228 – Functional model of SSF AE

11.3 Relations between SSF FSM and the CCF and maintenance functions

The primitive interface between the SSF FSM and the CCF/maintenance functions is an internal interface and is not subject to standardisation in IN CS-2. Nevertheless, this interface should be in line with the Basic Call State Model (BCSM) defined in 4.2/Q.1224.



CSA Call Segment Association

CS Call Segment

NOTE – One of the three possible FSMs (either IN-SSM, Assisting SSF or Handed-off SSF) is selected.

Figure 11-2/Q.1228 – SSF interfaces

An instance of an SSF FSM is either an IN Switching State Model (IN-SSM) FSM, or an Assisting SSF FSM or an Handed-off SSF FSM (see Figure 11-2).

The relationship between the BCSM and the SSF FSM is described as follows for the case of a call/attempt initiated by an end user, and the case of a call/attempt initiated by the SCF:

– When a call/attempt is initiated by an end user and processed at an exchange, a new instance of a BCSM is required. As the BCSM proceeds, it encounters Detection Points (DPs, see 4.2/Q.1224). If a DP is armed as a Trigger DP (TDP), an instance of an SSF FSM is required.

– If an InitiateCallAttempt is received from the SCF, an instance of a BCSM is created, as well as an instance of an SSF FSM.

The SSF logic shall:

– perform the DP processing actions specified in 4.2.8/Q.1224, including if DP criteria are met;

– check if traffic mechanisms are active;

– check for SCF accessibility;

– handle service feature interactions.

The SSF hands control back to the CCF at least in the following cases:

– if call gapping is in effect: the SSF logic instructs the CCF to terminate the call with the appropriate treatment;

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

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

– if the call is abandoned: the SSF logic returns call control to the CCF and continues processing as described in 11.5;

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

– if there is an existing control relationship for the call and a DP is encountered which is armed as an TDP-R: the SSF returns call control to the CCF.

The management functions related to the execution of operations received from the SCF are executed by the SSF Management Entity (SSME). The SSME comprises a SSME-Control and several instances of SSME FSMs. The SSME-control interfaces the different SSF FSMs and SSME FSMs respectively and the Functional Entity Access Manager (FEAM). Figure 11-2 shows the SSF Interfaces.

The Functional Entity Access Manager (FEAM) provides the low level interface maintenance functions including the following:

1) establishing and maintaining the interfaces to the SCF and SRF;

2) passing and queueing (when necessary) the messages received from the SCF and SRF to the SSME-Control;

3) formatting, queueing (when necessary), and sending the messages received from the SSME-Control to the SCF and SRF.

The SSME-control maintains the dialogues with the SCF, and SRF on behalf of all instances of the SSF Finite State Model (FSM). These instances of the SSF FSM occur concurrently and asynchronously as calls occur, which explains the need for a single entity that performs the task of creation, invocation, and maintenance of the SSF FSMs. In particular, the SSME-control performs the following tasks:

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

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

3) Captures asynchronous (with call processing) activities related to management or supervisory functions in the SSF and creates an instance of a SSME FSM. For example, the SSME provides non-call associated treatment due to changes in Service Filtering or Call Gapping. Therefore, the SSME-control separates the SSF FSM from the Call Gapping and Service Filtering functions by creating instances of SSME FSMs for each context of management related operations.

The different contexts of the SSME FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering this address information is given by filteringCriteria, i.e. all ActivateServiceFiltering operations using the same address, address the same SSME FSM handling this specific service filtering instance. For example,

ActivateServiceFiltering operations providing different filtering Criteria cause the invocation of new SSME FSMs.

The SSF FSM passes call handling instructions to the related instances of the BCSM as needed. DPs may be dynamically armed as Event DPs, requiring the SSF FSM to remain active. At some point, further interaction with the SCF is not needed, and the SSF FSM may be terminated while the BCSM continues to handle the call as needed. A later TDP in the BCSM may result in a new instance of the SSF FSM for the same call.

Consistent with the single-ended control characteristic of IN service features for IN CS-1, the SSF FSM only applies to a functionally separate call portion (e.g. the originating BCSM or the terminating BCSM in a two-party call, but not both).

11.4 SSF management finite state model (SSME FSM)

The SSME FSM state diagram is described in Figure 11-3.

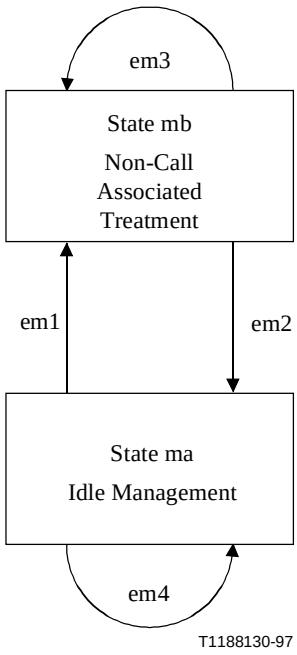


Figure 11-3/Q.1228 – SSME FSM state diagram

The SSME FSM is independent of the individual SSF FSMs.

In the Idle Management state, the following operations may be received from the SCF and processed by the SSME FSM with no resulting transition to a different state (transition em4):

RequestCurrentStatusReport

ActivityTest

ManageTriggerData

The ActivityTest operation applies to call context transactions only.

The ManageTriggerData operation can only be received outside a call context transaction.

The Non-Call Associated Treatment state is entered from the Idle Management state when one of the following non-call associated operations is received (transition em1):

RequestEveryStatusChangeReport

RequestFirstStatusMatchReport

ActivateServiceFiltering

CallGap

The CallGap operation can be received inside as well as outside a call context transaction. The ActivateServiceFiltering operation can be received outside a call context transaction only.

During this state, the following events can occur:

– given that service filtering is active, the SSF shall send a service filtering response to the SCF: the SSME FSM remains in this state (transition em3);

– given that service filtering is active, the SSF shall increment a counter; the SSME FSM remains in this state (transition em3);

– given that service filtering is active and the service filtering duration expires: the SSME shall send a **ServiceFilteringResponse** operation to the SCF; the SSME FSM moves to the Idle Management state (transition em2);

– given that status report is active, as previously requested by a RequestEveryStatusChangeReport operation, the SSF sends a **StatusReport** operation; the SSME FSM remains in this state (transition em3);

– given that status report is active, as previously requested by a RequestFirstStatusMatchReport operation, the SSF sends a **StatusReport** operation; the SSME FSM transits to Idle Management state (transition em2);

– given that status report is active, as previously requested by a RequestFirstStatusMatchReport or a RequestEveryStatusMatchReport operation, and the CancelStatusReportRequest operation is received by the SSF or the status report duration expires, the SSME FSM moves to the Idle Management state (transition em2);

– if call gap related duration timer expires, the SSME FSM moves to the Idle Management state (transition em2);

– given that call gap/service filtering is active, another CallGap/ActivateServiceFiltering operation having the same gapping/filtering criteria can be received by the SSF: the second "filter" or "gap" replaces the first one (transition em3) unless the duration timer value is equal to zero, in which case the SSME FSM moves to the Idle Management state (transition em2).

All other operations have no effect on the SSME FSMs; the operations are passed by the SSME-Control to the relevant SSF FSM.

11.5 SSF switching state model (SSM) FSM

The SSM FSM consists of a FSM for a Call Segment Association (FSM for CSA). The FSM for CSA creates one or more sub FSMs for Call Segment (FSM for CS).

NOTE – Within this subclause, the term *DP-specific operations* is used. DP-specific operations are the following:

AnalysedInformation

CollectedInformation

FacilitySelectedAndAvailable

OAbandon

OAnswer

OCalledPartyBusy

ODisconnect

OMidCall

ONoAnswer

OriginationAttempt

OriginationAttemptAuthorized

OSuspended

RouteSelectFailure

TAnswer

TBusy

TDisconnect

TermAttemptAuthorized

TerminationAttempt

TMidCall

TNoAnswer

TSuspended

General rules and procedure principles for inclusion of Call Party Handling (CPH) capabilities into the SSF FSMs are addressed here.

– Timer treatment:

  The use of a Timer to guard the SSF-SCF association (TC dialogue) or to prevent against excessive call suspension shall be done at the CS level.

– The change of a Connection View (CV) is initiated either from the End User side (e.g. midcall DP) or from the SCF (SCF initiated CV change).

– The SCF may change the CV by sending of one of the following operations:

  AnalyseInformation

  Connect

  ContinueWithArgument

  DisconnectLeg

  InitiateCallAttempt

  SplitLeg

  MergeCallSegments

  MoveCallSegments

  MoveLeg

  Reconnect

  ReleaseCall

  SelectRoute

– The SCF is informed about changes of the CV via following operations:

  EventReportBCSM, DP specific operations, EntityReleased.

– The SCF is informed about changes in the CV initiated by the SCF via CPH operations when ReturnResults are sent by the SSF on a successful change of the CV.

– The SCF can control a leg for which at least the Disconnect DP was armed. (Lifetime supervision of the leg.)

– The SCF shall have a connection view of the legs involved in the call. This is done by informing the SCF about the leg state changes, e.g. the disconnect of a leg. It shall not be allowed to have legs at a connection point which are not visible to the SCF (i.e. no DP armed).

– The number of operations that will resume call processing to be sent by the SCF in case the FSM for the CS is put into Waiting For Instructions state shall be equal to the number of events that causes/requested the suspension of the call process, i.e. each reported intercepted event to the SCF has to be responded by the SCF with an operation that has the request to resume call processing (e.g. ContinueWithArgument, Connect, AnalyseInformation). The FSM for CS has to keep track of the number of outstanding responses and await the transition in the FSM for CS to resume call processing (e.g. transition from Waiting For Instructions to Monitoring) until all outstanding responses have been catered for.

– CPH procedure principles for the FSM for CS:

  • The import of a leg (including EDPs or pending reports) in a CS (target CS) shall not affect the processing for other leg(s) residing in the same CS.

  • The FSM for CS (one FSM per Call Segment and Connection Point) shall not know how many legs are connected to the CS. The FSM for CS exists as long as at least one report is pending or a DP is armed.

  • In the states Waiting for End of User Interaction and Waiting for End of Temporary Connection the following operations are not allowed:

    DisconnectLeg

    SplitLeg

    MoveLeg

    MergeCallSegments

    MoveCallSegments

    Reconnect

    ContinueWithArguments

  • Per one CS only one connection is allowed to a resource (i.e. User Interaction or Temporary Connection).

  • All CPH operations received by the FSM for CS shall cause a transition to the state Waiting For Instructions:

    – If one of the following operations DisconnectLeg, SplitLeg, MoveLeg or MergeCallSegments is received in the Monitoring state, the FSM for CS shall process the received operation and then it shall transit to the Waiting for Instructions state (valid for SCF initiated changes of Connection view states).

    – The receipt of one of the above operations in the state Waiting for Instructions shall not cause the change of the state Waiting for Instructions. Therefore all operation sequences shall be finalised by an operation which changes the state into Monitoring (e.g. ContinueWithArgument, Connect), except the case that an operation causes the transition to state Idle. (e.g. DisconnectLeg for the last leg.)

    – When the SplitLeg operation is received in the Idle state for the target FSM for CS, the FSM for CS shall process the received operation and shall then transit to the Waiting for Instructions state.

Each FSM and the according states are discussed in the following subclauses. General rules applicable to more than one FSM/state are addressed here.

One or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and is processed as follows:

– Process the operations in the order in which they are received.

– Each operation causes a state transition independent of whether or not a single operation or multiple operations are received in a message.

– The SSF examines subsequent operations in the sequence. As long as sequential execution of these operations leaves the FSM in the same state, it shall execute them (e.g. RequestReportBCSMEvent). If a subsequent operation causes a transition out of the state, then the following operations shall be buffered until the current operation has been executed. In all other cases, await an event that causes a transition out of the current state (such an event is the completion of operation being executed, or the reception of an external event). An example of this is as follows:

  The SSF receives the operations FurnishChargingInformation, ConnectToResource, and PlayAnnouncement in a component sequence inside a single TCAP message. Upon receipt of this message, these operations are executed up to and including ConnectToResource while the SSF is in the Waiting For Instructions state. As the ConnectToResource operation is executed (and when, or after the FurnishChargingInformation operation has been completed), the SSF FSM will transit to the Waiting For End Of User Interaction state. The PlayAnnouncement operation is relayed to the SRF while the SSF is in Waiting For End Of User Interaction state.

– If there is an error in processing one of the operations in the sequence, the SSF FSM processes the error (see below) and discards all remaining operations in the sequence.

– If an operation is not understood or is out of context (i.e. violates the SACF rules defined by the SSF FSM) as described above, ABORT the interaction by sending of a TCAP ABORT at the CSA level or an operation EntityReleased at the CS level. For example, when the FSM for CS applies to an originating BCSM then receiving SelectFacility operation would be out of context since this applies only to the terminating half of the BCSM.

In any state, if there is an error in a received operation, the maintenance functions are informed and the SSF FSM remains in the same state as when it received the erroneous operation; depending on the class of the operation, the error could be reported by the SSF to the SCF using the appropriate component (Recommendation Q.774).

11.5.1 Finite State Model for Call Segment Association (CSA)

Figure 11-4 shows the state diagram of the FSM for CSA in the SSF part of the SSP during the processing of an IN call/attempt.

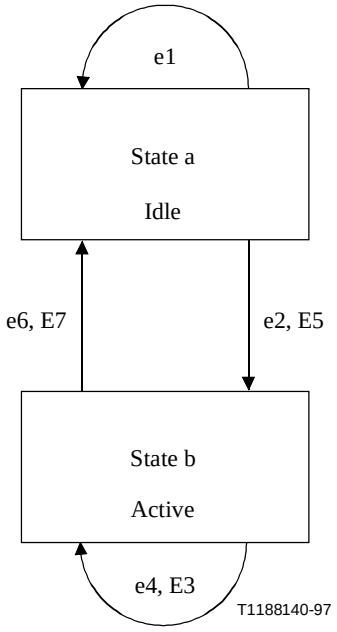


Figure 11-4/Q.1228 – FSM for Call Segment Association

An instance of the FSM for CSA is created by SSME-Control when:

– an indication of a new call attempt is received from a user;

– a message related to a new transaction containing an InitiateCallAttempt or CreateCallSegmentAssociation operation is received from the SCF.

The FSM for CSA state diagram contains the following transitions (events):

e1 – TDP-N encountered.

e2 – TDP-R encountered.

E3 – Any operation received from SCF except transit to state Idle (includes InitiateCallAttempt).

e4 – EDP-R encountered.

– EDP-N last encountered except last one from last CS.

– any response and report.

E5 – InitiateCallAttempt (received in state Idle).

– CreateCallSegmentAssociation received.

e6 – last EDP-N from last CS and no other reports pending.

E7 – any operation received from SCF which causes no remaining CS (e.g. ReleaseCall, MoveCallSegments or DisconnectLeg).

The CSA state diagram contains the following states:

State a: Idle.

State b: Active.

11.5.1.1 State a: Idle

The FSM for CSA enters the Idle state under a variety of conditions, as described below.

The FSM for CSA enters the Idle state when sending or receiving an ABORT TCAP primitive due to abnormal conditions in the state Active.

The FSM for CSA enters the Idle state when one of the following occurs:

– when all the FSM for Call Segment instances associated with the FSM for CSA instance are released.

During this state, the following call-associated events can occur:

– The following operation may be received from the FSM for CS with no resulting transition to another state (transition e1):

  • an **InitialDP** or **DP-specific operation** (see Note) is received indicating an encountered TDP-N. The received operation is sent to the SCF.

– The following operation may be received from the FSM for CS, causing a state transition to the Active state (transition e2):

  • an **InitialDP** or **DP-specific operation** (see Note) is received indicating an encountered TDP-R. The received operation is sent to the SCF.

  The rules for DP processing are described in 4.2.8/Q.1224, "DP Processing".

– a message related to a new transaction containing an **InitiateCallAttempt** or a **CreateCallSegmentAssociation** operation is received from the SCF: in this case the FSM for CSA moves to the state Active (transition E5).

Any other operation received from the SCF while the FSM for CSA is in Idle state, i.e. while the SSF FSM instance does not exist, should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TC (refer to clause 18).

NOTE – For a definition of *DP-specific operations,* see the beginning of 11.5.

11.5.1.2 State b: Active

This state is entered from the Idle state by detecting a TDP-R (transition e2), from the Idle state on receipt at the SSF of a TC_Begin indication primitive containing an **InitiateCallAttempt** or a **CreateCallSegmentAssociation** operation from the SCF (transition E5).

In this state the FSM for CSA handles instructions from the SCF and events which are received from the FSMs for CS.

During this state, the following events can occur:

– The receipt of an END or ABORT primitive from TCAP has no effect on the call; the call may continue or be completed with the information available. In this case, the FSM for CSA transits to the Idle state (transition E7), disassociating the FSM for CSA from the call.

– An operation is received from the SCF: The FSM for CSA acts according to the operation received as described below.

– An operation is received from the FSM for CS: The FSM for CSA acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to the idle state (transition E3):

ApplyCharging

CallInformationRequest

Cancel(invokeID)

ConnectToResource

DisconnectForwardConnection

DisconnectForwardConnectionWithArgument

EstablishTemporaryConnection

FurnishChargingInformation

NOTE 1 – It is network-operator specific whether the operation is forwarded to any FSM for CS instance using the internal coding of the OCTET STRING or not.

HoldCallInNetwork

InitiateCallAttempt

NOTE 2 – In this case the FSM for CSA creates a new FSM for CS instance and transmits the event to it.

MergeCallSegments

NOTE 3 – In this case, the SSF deletes the "source" Call Segment and connects the Leg in the "source" Call segment with the "target" Call Segment. The FSM for CSA transmits the event to the FSM instance for the "source" CS and releases the FSM instance. Furthermore, the FSM for CSA transmits the event to the FSM instance for the "target" CS.

MoveCallSegments (for target CS)

MoveLeg

NOTE 4 – In this case, the SSF moves the leg from the "source" Call Segment to the "target" Call Segment with which the source Call Segment is associated. The FSM for CSA transmits the event to the 'source' FSM for CS instance and to the "target" FSM for CS instance.

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

RequestNotificationChargingEvent

RequestReportFacilityEvent

ResetTimer

ScriptClose

ScriptInformation

ScriptRun

SendChargingInformation

SendFacilityInformation

SplitLeg

NOTE 5 – In this case, the SSF creates a Call Segment and connects the split Leg with the Call Segment. The FSM for CSA transmits the event to the FSM instance for the "source" CS. Furthermore, the FSM for CSA creates a new FSM instance for the 'target' Call Segment and transmits the event to the FSM.

The following operations may be received from the SCF, causing a state transition either to the same state if an FSM for CS instance exists after this event was processed in the FSM (transition E3), or to the Idle state if all the FSM for CS instances associated with the FSM for CSA instance transit to the Idle state (transition E7).

AnalyseInformation

AuthorizeTermination

Cancel(allRequests)

CollectInformation

Connect

Continue

ContinueWithArgument

DisconnectLeg

InitialDP

MoveCallSegments (for source CS)

Reconnect

RequestReportBCSMEvent

SelectFacility

SelectRoute

The **ReleaseCall** operation may be received from the SCF. For the case the whole CSA is to be released, the FSM for CSA shall instruct all the relevant FSM for CS instances to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

– if the last CS has been released, and if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CSA transits to the Idle state (transition E7);

– if the last CS has been released, and if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CSA transits to the Idle state (transition E7);

CSs are created or deleted by the CSA, if necessary, then the operations are passed to the appropriate CS and are processed there.

The following operations may be received from one of the CS, causing a state transition either to the same state if an FSM for CS instance exists after this event was processed in the FSM (transition e4), or to the Idle state if all the FSM for CS instances associated with the FSM for CSA instance transit to the Idle state (transition e6). The operations are sent to the SCF:

ApplyChargingReport

CallInformationReport

DPSpecific operations (for EDP)

EntityReleased

EventReportBCSM

EventReportFacility

The following operations may be received from one of the CS with no resulting transition to the idle state (transition e4), the operations are sent to the SCF:

EventNotificationCharging

ReturnResult for PromptAndCollectUserInformation

ReturnResult for PromptAndReceiveMessage

ScriptEvent

SpecializedResourceReport

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.5.2 Finite State Model for Call Segment

See Figure 11-5.

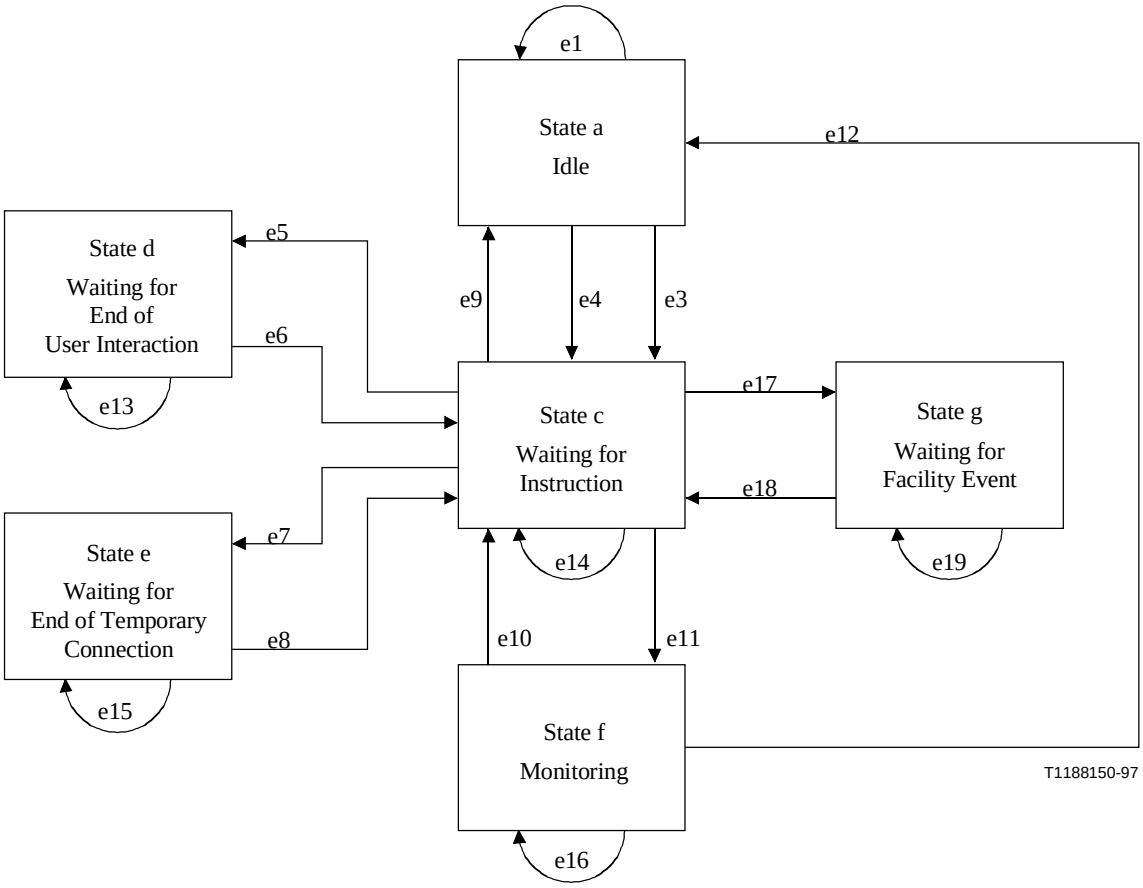


Figure 11-5/Q.1228 – FSM for Call Segment

The SSF state diagram for the CS contains the following transitions (events):

- e1 – TDP-N encountered.
- e3 – InitiateCallAttempt received, SplitLeg received (transition when "target" CS).
- e4 – TDP-R encountered.
- e5 – User interaction requested.
- e6 – User interaction ended.
- e7 – Temporary connection created.
- e8 – Temporary connection ended.
- e9 – Idle return from Waiting for Instructions.
- e10 – EDP-R encountered.
- e11 – Routing instruction received.
- e12 – EDP-N last (see Note 1) encountered or ReleaseCall received or Cancel(allRequests) received.
- e13 – Waiting For End Of User Interaction state no change.
- e14 – Waiting For Instructions state no change.
- e15 – Waiting For End Of Temporary Connection state no change.

e16 – Monitoring state no change.

e17 – Request Report Facility Event has been received.

e18 – Event Report Facility (last armed Facility Event case) has been issued.

e19 – Waiting For Facility Event state no change.

NOTE 1 – The "last EDP-N" means that there are no other EDPs which may be encountered when an EDP-N was detected. Some of the EDPs are automatically disarmed if another EDP is encountered. The EDPs which are automatically disarmed depend on which EDP is encountered. An example is the case of the EDPs O_Answer, O_No_Answer, RouteSelectFailure or O_Called_Party_Busy. If any of these EDPs are encountered, all the other EDPs of this list are automatically disarmed.

NOTE 2 – The Abandon and Disconnect events can be encountered in any state; the transition to the next state is dependent on the current state of call processing.

NOTE 3 – Non-Call Associated Treatment events (refer to 11.4) can be encountered in any state, the FSM for CS remains in the same state.

The SSF state diagram contains the following states:

State a: Idle.

State c: Waiting For Instructions.

State d: Waiting For End Of User Interaction.

State e: Waiting For End Of Temporary Connection.

State f: Monitoring.

State g: Waiting For Facility Event.

In any state (except Idle), if the calling party abandons the call before it is answered (i.e. before the Active PIC in the BCSM), then the FSM for CS instance shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

– If the Abandon DP is not armed and there is no CallInformationReport pending, then transit to the Idle state.

– If the Abandon DP is not armed and there is a CallInformationReport pending, send a **CallInformationReport**, then transit to the Idle state.

– If the Abandon DP is armed as an EDP-R, send an **EventReportBCSM** or a **DP-specific operation**, then transit to the Waiting for Instructions state. If a CallInformationReport is pending, a **CallInformationReport** shall be sent before the corresponding **EventReportBCSM** or a DP-specific operation is sent.

– If the Abandon DP is armed as an EDP-N and there is no **CallInformationReport** pending, send an **EventReportBCSM** or a **DP-specific operation**, then transit to the Idle state.

– If the Abandon DP is armed as an EDP-N and there is a CallInformationReport pending, send an **EventReportBCSM** or a **DP-specific operation**, followed by a **CallInformationReport**, then transit to the Idle state.

– Other pending requests that should be treated in the same way as the CallInformationReport in the above list is the **ApplyCharging**.

In any state (except Idle), if a call party disconnects from a stable call (i.e. from the Active PIC in the BCSM), then the SSF FSM shall process this event as follows:

– If the Disconnect DP is not armed for that specific leg and there is no CallInformationReport pending, transit to the Idle state.

– If the Disconnect DP is not armed and there is a CallInformationReport pending, send a **CallInformationReport** and transit to the Idle state.

– If the Disconnect DP is armed as an EDP-R for that specific leg, send an **EventReportBCSM** or a **DP-specific operation**, then transit to the Waiting For Instructions state. If a CallInformationReport is pending, a **CallInformationReport** shall be sent before the corresponding EventReportBCSM or a DP-specific operation is sent.

– If the Disconnect DP is armed as an EDP-N and there is no CallInformationReport pending, send an **EventReportBCSM** or a **DP-specific operation**, then transit to the Idle state.

– If the Disconnect DP is armed as an EDP-N and there is a CallInformationReport pending, send an **EventReportBCSM** or a **DP-specific operation** and a **CallInformationReport**, then transit to the Idle state.

– Other pending requests that should be treated in the same way as the CallInformationReport in the above list is the **ApplyCharging**.

In any state (except Idle), an **EventNotificationCharging** can be sent to the SCF, if previously requested by a RequestNotificationChargingEvent and if the charging event has been detected by the CCF. In this case, no state transition takes place.

Each FSM for CS instance has an application timer, T_{SSF}, whose purpose is to prevent excessive call suspension time and to guard the association between the SSF and the SCF.

If necessary, the Timer T_{SSF} may be set in the following cases:

– when the SSF sends an InitialDP or a DP specific operation for a TDP-R (see 11.5.2.2, State c: Waiting For Instructions). While waiting for the first response from the SCF, the timer T_{SSF} can be restarted only once by a ResetTimer operation. Subsequent to the first response, the timer can be restarted any number of times.

– when the FSM for CS enters the Waiting For Instructions state (see 11.5.2.2) under any other condition than the ones listed in the previous case. In this case, the SCF may restart the T_{SSF} timer using the ResetTimer operation any number of times.

– when the FSM for CS receives a HoldCallInNetwork operation (see 11.5.2.2, State c: Waiting For Instructions). In this case, the SCF may restart T_{SSF} using the ResetTimer operation any number of times.

– when the SSF enters the Waiting For End Of User Interaction state or the Waiting For End Of Temporary Connection state (see 11.5.2.3 and 11.5.2.4). In these cases, the SCF may restart T using the ResetTimer operation any number of times. (OPTIONAL.)

NOTE 4 – This "OPTIONAL" means that the application timer T_{SSF} is optionally set. Whether it is used or not depends on implementation. But it must be synchronised with $T_{SCF\text{-}SSF}$ in the SCSM.

In each of the above cases, T_{SSF} may have different values as defined by the application.

When receiving or sending any operation which is different from the above, the FSM for CS instance shall restart T_{SSF} with the last used value. This value is either one associated to the different cases as listed above, or received in a ResetTimer operation, whatever occurred last. In the Monitoring state (see 11.5.2.5), T_{SSF} is not used.

On expiration of T_{SSF}, the FSM for CS transits to the Idle state and the CCF progresses the BCSM if possible. If the FSM for CS was the last in the CSA the interaction with the SCF is aborted, otherwise the operation EntityReleased is sent to the SCF.

11.5.2.1 State a: Idle

The FSM for CS enters the Idle state under a variety of conditions, as described below.

The FSM for CS enters the Idle state when the associated FSM for CSA instance transits to the Idle state.

The FSM for CS enters the Idle state when one of the following occurs:

– when the call is abandoned or one or more call parties disconnect in any other state under the conditions identified in 11.5.2;

– when one of the following operations is processed in the Waiting For Instructions state, and no EDPs are armed and there are no outstanding report requests (transition e9);

AnalyseInformation

AuthorizeTermination

CollectInformation

Connect

Continue

(Only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multicall segment CSA.)

Reconnect

SelectFacility

SelectRoute

– when the application timer T_{SSF} expires in one of the states: Waiting For Instructions, Waiting For End Of User Interaction, Waiting For End Of Temporary Connection or Waiting For Facility Event;

– when a **ReleaseCall** operation is processed in Waiting For Instructions (transition e9) or Monitoring (transition e12);

– when the last EDP-N is encountered in the Monitoring state, there are no EDP-Rs armed and no outstanding report requests are to be monitored (transition e12);

– when the last charging event which was previously requested by the ApplyCharging operation is encountered in the Monitoring state, and there are no EDPs armed (transition e12);

– when a **MergeCallSegments** operation is received for the source segment (transition e9 or transition e12);

– when a **MoveLeg** operation is received for the last leg in the source call segment (transition e9 or transition e12);

– when a **ContinueWithArgument** operation is received, and there are no EDPs armed and no pending requests (transition e9);

– when a **DisconnectLeg** operation is received, and the last leg in the call segment is released (transition e9 or transition e12);

– when a **Cancel**(allRequests) operation is received in the Monitoring state (transition e12);

– when all the EDPs are disarmed in the Monitoring state, and there is no outstanding report (transition e12).

When transiting to the Idle state, if there is a CallInformationReport pending (see 11.5.2), the SSF sends a CallInformationReport operation to the SCF before returning to Idle. Once in the Idle state, if status reporting is still active the SSF deactivates it, any outstanding responses to send to the SCF are discarded.

During the state Idle, the following call-associated events can occur:

– indication from the CCF that an armed TDP-R is encountered related to a possible IN call/service attempt, the FSM for CS instance sends a generic **InitialDP** or **DP-specific operation** (see Note) to the associate FSM for CSA, as determined from DP processing, and transits to the Waiting For Instructions state (transition e4);

– indication from the CCF that an armed TDP-N is encountered related to a possible IN call/service attempt, the FSM for CS instance sends a generic **InitialDP** or **DP-specific operation** (see Note) to the associate FSM for CSA, as determined from DP processing, and transits to the Idle state (transition e1);

  The rules for DP processing are described in 4.2.8/Q.1224, "DP Processing".

– an **InitiateCallAttempt** operation is received from the SCF: in this case, a new FSM for CS instance is created by the associate FSM for CSA. This FSM for CS instance transits to the Waiting For Instructions state (transition e3).

– a **SplitLeg** operation is received from the SCF: in this case, a new target FSM for CS instance is created by the associate FSM for CSA. This FSM for CS instance transits to the Waiting For Instructions state (transition e3).

Any other operation received from the SCF while the FSM is in Idle state should be treated as an error. The event should be reported to the maintenance functions.

NOTE – For a definition of *DP-specific operations,* see the beginning of 11.5.

11.5.2.2 State c: Waiting For Instructions

This state is entered:

– from the Idle state, either directly as indicated above (transition e4), or on receipt of an **InitiateCallAttempt** or **SplitLeg** ("target" CS) operation from the SCF (transition e3);

– from the state Monitoring on detection of an EDP-R (transition e10);

– from the state Waiting For End Of User Interaction on occurrence of disconnection of the SRF (transition e6);

– from the state Waiting For End Of Temporary Connection on occurrence of disconnection of temporary connection (transition e8); or

– from the state Waiting For Facility Event on receiving of a Facility Event (last armed Facility event) (transition e18).

In this state the FSM for CS is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) shall be set on entering this state.

During the state Waiting For Instructions, the following events can occur:

– The user dials additional digits (applies for open-ended numbering plans): the CCF should store the additional digits dialled by the user.

– The user abandons or disconnects. This should be processed in accordance with the general rules in 11.5.

– The application Timer T_{SSF} expires: the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions, and the operation EntityReleased is sent to the FSM for CSA.

– The user issues a feature activation or a Switch Hook Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the Midcall DP was armed to be reported in any state except Idle.

– An operation is received from the SCF: The FSM for CS instance acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e14):

ApplyCharging

CallInformationRequest

Cancel(allRequests)

DisconnectLeg (for not last leg)

FurnishChargingInformation

HoldCallInNetwork

InitiateCallAttempt (for non initial CS)

MergeCallSegments (at target CS)

MoveLeg (for target CS and not last leg in source CS)

RequestNotificationChargingEvent

RequestReportBCSMEvent

ResetTimer

SendChargingInformation

SendFacilityInformation

SplitLeg

For the case where a **SplitLeg** or an **InitiateCallAttempt** (for non initial CS) operation is received from the SCF, a new CS and a new instance of an FSM for CS instance is created by the associate FSM for CSA. This new FSM instance shall receive the **SplitLeg** or **InitiateCallAttempt** in the Idle state and transit to the Waiting For Instructions state (transition e3). The FSM for CSA also sends the **SplitLeg** operation to the source FSM for CS instance, which may be in the Waiting for Instructions or Monitoring state.

The following operation may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of User Interaction state (transition e5):

ConnectToResource

The following operation may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of Temporary Connection state (transition e7):

EstablishTemporaryConnection

The following operations may be received from the SCF and processed by the SSF, causing a state transition to either Monitoring state (if any EDPs were armed or any reports were requested) (transition e11) or Idle state (transition e9):

AnalyseInformation

AuthorizeTermination

CollectInformation

Connect

Continue

(only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multicall segment CSA.)

ContinueWithArgument

Reconnect

SelectRoute

SelectFacility

The following operation may be received from the SCF and processed by the SSF, causing a state transition to Waiting For Facility Event state (transition e17):

RequestReportFacilityEvent

ReleaseCall (for a CS) or **DisconnectLeg** (for the last leg in the CS) operation may be received from the SCF. In this case, the FSM for CS instance shall instruct the CCF to clear the call segment and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

– if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CS transits to the Idle state (transition e9);

– if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CS transits to the Idle state (transition e9).

The **MergeCallSegments** (for "source" CS) operation or **MoveLeg** (for last leg in "source" CS) operation may be received from the SCF. In this case, the SSF FSM related to the "source" CS shall return to the Idle state (transition e9).

When processing the above operations, any necessary call handling information is provided to the Call Control Function (CCF).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.5.2.3 State d: Waiting For End Of User Interaction

The SSF enters this state from the Waiting For Instructions state (transition e5) on the reception of the operation **ConnectToResource**.

The timer T_{SSF} is active in this state. (whether it is used or not is optional.)

During this state, the following events can occur:

– One of the following valid SCF-SRF operations for relaying is received and is correct, the operation is transferred to the SRF for execution:

Cancel(invokeID)

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptClose

ScriptInformation

ScriptRun

The SSF FSM remains in the Waiting For End Of User Interaction state (transition e13).

– One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF:

SpecializedResourceReport

ReturnResult from PromptAndCollectUserInformation

ReturnResult from PromptAndReceiveMessage

ScriptEvent

The SSF FSM remains in the Waiting For End Of User Interaction state (transition e13).

– The application timer T_{SSF} expires (if it was set) : the SSF FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.

– The user issues a feature activation or a Switch Hook Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the Midcall DP was armed to be reported in any state except Idle.

– An operation is received from the SCF: The FSM for CS acts according to the operation received as described below.

– The user abandons. This should be processed in accordance with the general rules in 11.5.2.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e13):

ApplyCharging

FurnishChargingInformation

RequestNotificationChargingEvent

ResetTimer

SendChargingInformation

The **DisconnectForwardConnection** (only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multicall segment CSA) or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases, this causes the release of the connection to the SRF and the transition to the Waiting For Instructions state. The disconnection is not transferred to the other party (transition e6).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.5.2.4 State e: Waiting For End Of Temporary Connection

The FSM for CS enters this state from the Waiting For Instructions state (transition e7) upon receiving an **EstablishTemporaryConnection** operation.

The call is routed to the assisting SSF/SRF and call handling is suspended while waiting for the end of the assisting procedure. The timer T_{SSF} is active in this state. (whether it is used or not is optional.)

During the state Waiting For End Of Temporary Connection, the following events can occur:

– The applications timer T_{SSF} expires (if it was set): the SSF FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.

– The user issues a feature activation or a Switch Hook Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the Midcall DP was armed to be reported in any state except Idle.

– The receipt of an indication of disconnection of forward connection from the CCF. In this case, the SSF moves to the Waiting For Instructions state (transition e8). The disconnection is not transferred to the calling party.

– The user abandons. This should be processed in accordance with the general rules in 11.5.2.

– An operation is received from the SCF; the FSM for CS acts according to the operation received as described below.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e15):

ApplyCharging

FurnishChargingInformation

RequestNotificationChargingEvent

ResetTimer

SendChargingInformation

The **DisconnectForwardConnection** (only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multicall segment CSA) or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases, this causes the release of the connection to the SRF and the transition to the Waiting For Instructions state. The disconnection is not transferred to the other party (transition e8).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.5.2.5 State f: Monitoring

The FSM for CS enters this state from the Waiting For Instructions state (transition e11) upon receiving one of the following operations when one or more EDPs are armed or/and there are other reports pending (see 11.5):

AnalyseInformation

AuthorizeTermination

Connect

Continue

(Only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multicall segment CSA.)

ContinueWithArgument

CollectInformation

Reconnect

SelectRoute

SelectFacility

The timer T_{SSF} is stopped on entering this state.

During the state Monitoring, the following events can occur:

– An EDP-N is reported to the SCF by sending an **EventReportBCSM** operation or a **DP-specific operation**; the FSM for CS shall remain in the Monitoring state (transition e16) if one or more EDPs are armed or there are report requests pending. The FSM for CS shall move to the Idle state (transition e12) if there are no remaining EDPs armed or there are no report requests pending.

  If the event causing a CallInformationReport is also detected by an armed EDP-N, then the **CallInformationReport** shall be sent immediately after the corresponding **EventReportBCSM** or a **DP-specific operation** is sent.

– An EDP-R should be reported to the SCF by sending an **EventReportBCSM** operation or a **DP-specific operation**; the SSF FSM should move to the Waiting For Instructions state (transition e10).

  If the event causing a CallInformationReport is also detected by an armed EDP-R, then the **CallInformationReport** shall be sent immediately before the corresponding **EventReportBCSM** or a **DP-specific operation** is sent.

– An operation is received from the SCF: The FSM for CS acts according to the operation received as described below.

– The user abandons or disconnects. This should be processed in accordance with the general rules in 11.5.2.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e16):

ApplyCharging

FurnishChargingInformation

Reconnect

RequestNotificationChargingEvent

SendChargingInformation

The following operations may be received from the SCF and processed by the SSF, causing a state transition to the Waiting For Instructions state (transition e10, valid for SCF initiated changes of Connection view states):

DisconnectLeg (for not the last leg)

MergeCallSegments (for target CS)

MoveLeg (for target CS and not the last leg in source CS)

SplitLeg

The following operations may be received from the SCF and processed by the SSF, causing a state transition to the Idle state (transition e12, valid for SCF initiated changes of Connection view states):

DisconnectLeg (for the last leg)

MergeCallSegments (for source CS)

MoveLeg (for the last leg in source CS)

The **RequestReportBCSMEvent** operation may be received from the SCF. In this case, the FSM for CS transits as follows:

– if one or more EDPs are armed or a part of armed EDPs are disarmed, the FSM for CS transits back to the same state (transition e16);

– if all of armed EDPs are disarmed and there is outstanding CallInformationReport, EventNotificationCharging or ApplyChargingReport, the FSM for CS transits back to the same state (transition e16);

– if all of armed EDPs are disarmed and there is no outstanding CallInformationReport, EventNotificationCharging or ApplyChargingReport, the FSM for CS transits to the Idle (transition e12).

The **Cancel**(allRequests) operation received from the SCF shall be processed by the FSM for CS, causing a state transition to the Idle state (transition e12).

When the **ReleaseCall** operation is received from the SCF, the FSM for CS shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

– if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CS transits to the Idle state (transition e12);

– if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CS shall transit to the Idle state (transition e12).

During the Monitoring state, it is possible to perform user interaction in order to send tones, announcements and display information.

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.5.2.6 State g: Waiting For Facility Event

The FSM for CS enters this state from the Waiting For Instructions state (transition e17) upon receiving a **RequestReportFacilityEvent** operation from the SCF.

This state is equivalent to the state Waiting For Instructions, i.e. the FSM for CS is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) shall be restarted on entering this state; but in addition the SCF is waiting for an outstanding Facility Events.

During the state Waiting For Facility Event, the following events can occur:

– An Facility event is reported to the SCF by sending an **EventReportFacility** operation. The FSM for CS shall remain in the Waiting For Facility Event state (transition e19) if one or more Facility events are still outstanding. The FSM for CS shall move to the Waiting For Instructions state (transition e18) if there are no remaining Facility events outstanding (last armed Facility Event case).

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e19):

ApplyCharging

CallInformationRequest

Cancel(allRequests)

DisconnectLeg (for not last leg)

FurnishChargingInformation

HoldCallInNetwork

MergeCallSegments (at target CS)

MoveLeg (for target CS and not last leg in source CS)

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportFacilityEvent

ResetTimer

SendChargingInformation

SendFacilityInformation

SplitLeg

If an operation which resumes the call processing is received in this state (e.g. Continue, ContinueWithArgument, Connect), it shall be handled as an error.

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.6 Assisting SSF FSM

This subclause describes the SSF FSM related to the Assisting SSF.

The Assisting SSF state diagram contains the following transitions (events) (see Figure 11-6):

ea1 – Assist detected.

ea2 – Assist failed.

ea3 – User interaction requested.

ea4 – User interaction ended.

ea5 – Waiting For Instructions state no change.

ea6 – Waiting For End Of User Interaction state no change.

ea7 – Idle Return from Waiting For End Of User Interaction.

The Assisting SSF state diagram contains the following states:

State aa: Idle.

State ab: Waiting For Instructions.

State ac: Waiting For End Of User Interaction.

11.6.1 State aa: Idle

The FSM for Assisting SSF enters the Idle state when one of the following occurs:

– when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state;

– given a temporary connection between an Initiating SSF and the Assisting SSF, when a bearer channel disconnect is received from the initiating SSF; (transition ea2).

Once in the Idle state, if there are any outstanding responses to send to the SCF, they are discarded by the Assisting SSF.

The FSM for Assisting SSF transits from the Idle state to the Waiting For Instructions state on receipt of an assist indication at the assisting SSF from another SSF (transition ea1).

Any operation received from the SCF while the Assisting SSF is in Idle state shall be treated as an error. The event shall be reported to the maintenance functions and the transaction shall be aborted according to the procedure specified in TCAP (see Recommendation Q.774).

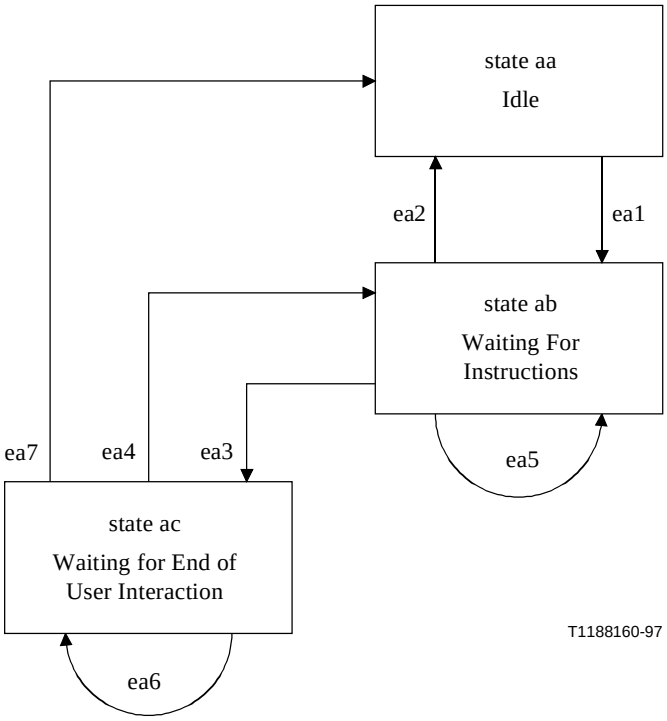


Figure 11-6/Q.1228 – FSM for Assisting SSF

11.6.2 State ab: Waiting For Instructions

This state is entered from the Idle state on receipt of a connect message at an SSF from another SSF indicating that an assist is required, based on an implementation dependent detection mechanism (transition ea1).

Before entering this state, the SSF sends an **AssistRequestInstructions** operation to the SCF and the FSM for Assisting SSF is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) shall be set on entering this state.

During this state, the following events can occur:

– The application timer T_{SSF} expires: the FSM for Assisting SSF moves to the Idle state (transition ea2) and the expiration is reported to the maintenance functions and the transaction is aborted.

– An operation is received from the SCF: The FSM for Assisting SSF acts according to the operation received as described below.

– A bearer channel disconnect is received and the FSM moves to the Idle state (transition ea2).

The following operations can be received from the SCF and processed by the Assisting SSF with no resulting transition to a different state (transition ea5):

ApplyCharging

FurnishChargingInformation

ResetTimer

SendChargingInformation

The following operation can be received from the SCF and processed by the Assisting SSF, causing a state transition to Waiting For End Of User Interaction state (transition ea3):

ConnectToResource

In the case where an implementation is not capable of differentiating between a Handed-off and an Assisting SSF case, it may execute the **ReleaseCall** operation in the assisting SSF. The FSM for Assisting SSF shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

– if ApplyChargingReport operation is not requested, the FSM for Assisting SSF transits to the Idle state (transition ea2);

– if ApplyChargingReport operation has been requested, the FSM for Assisting SSF sends **ApplyChargingReport** to SCF and then the FSM for Assisting SSF transits to the Idle state (transition ea2).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.6.3 State ac: Waiting For End Of User Interaction

The FSM for Assisting SSF enters this state from the Waiting For Instructions state (transition ea3) on the reception of the following operation:

ConnectToResource

During this state, the following events can occur:

– One of the following valid SCF-SRF operations for relaying is received and is correct, the operation is transferred to the SRF for execution:

Cancel(invokeID)

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptClose

ScriptInformation

ScriptRun

The FSM for Assisting SSF remains in the Waiting For End Of User Interaction state (transition ea6).

– One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF:

SpecializedResourceReport

ReturnResult from PromptAndCollectUserInformation

ReturnResult from PromptAndReceiveMessage

ScriptEvent

The SSF for Assisting SSF remains in the Waiting For End Of User Interaction state (transition ea6).

– The application timer T_{SSF} expires: the FSM for Assisting SSF moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted (transition ea7).

– An operation is received from the SCF: The FSM for Assisting SSF acts according to the operation received as described below.

– A bearer channel disconnect is received from the initiating SSF and the FSM moves to the Idle state (transition ea7).

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition ea6):

ResetTimer

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the Assisting SSF in this state, causing a transition to the Waiting For Instructions state (transition ea4). This procedure is only valid if a ConnectToResource was previously processed to cause a transition into the Waiting For End Of User Interaction state.

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.7 Handed-off SSF FSM

This subclause describes the SSF FSM related to the Handed-off SSF. The SSF FSM for IN CS-2 applies only to the case where final treatment is to be applied.

The Handed-off SSF state diagram contains the following transitions (events) (see Figure 11-7):

eh1 – Handed-off detected.

eh2 – Handed-off fail.

eh3 – User interaction requested.

eh4 – User interaction ended.

eh5 – Waiting For Instructions state no change.

eh6 – Waiting For End Of User Interaction state no change.

eh7 – Idle Return from Waiting For End Of User Interaction.

The Handed-off SSF state diagram contains the following states:

State ha: Idle.

State hb: Waiting For Instructions.

State hc: Waiting For End Of User Interaction.

11.7.1 State ha: Idle

The FSM for Handed-off SSF enters the Idle state when one of the following occurs:

– when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state.

When the bearer channel disconnects, FSM for Handed-off FSM shall also move to Idle.

Once in the Idle state, if there are any outstanding responses to send to the SCF, they are discarded by the Handed-off SSF.

The FSM for Handed-off SSF transits from the Idle state to the Waiting For Instructions state on receipt of an assist indication at the Handed-off SSF from another SSF (transition eh1).

Any operation received from the SCF while the Handed-off SSF is in Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TCAP (see Recommendation Q.774).

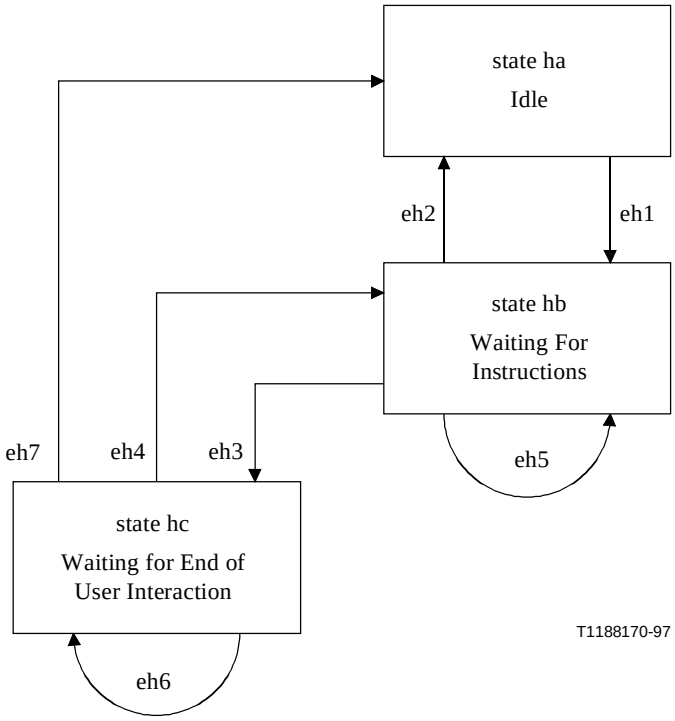


Figure 11-7/Q.1228 – FSM for Handed-off SSF

11.7.2 State hb: Waiting For Instructions

This state is entered from the Idle state on receipt of a connect at an SSF from another SSF indicating that a hand-off is required, based on an implementation dependent detection mechanism (transition eh1).

Before entering this state, the SSF sends an AssistRequestInstructions operation to the SCF and the FSM for Handed-off SSF is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) should be set on entering this state.

During this state, the following events can occur:

– The application timer T_{SSF} expires: the FSM for Handed-off SSF moves to the Idle state (transition eh2) and the expiration is reported to the maintenance functions and the transaction is aborted.

– An operation is received from the SCF: The FSM for Handed-off SSF acts according to the operation received as described below.

– A bearer channel disconnect is received and the FSM moves to the Idle state (transition eh2).

The following operations may be received from the SCF and processed by the Handed-off SSF with no resulting transition to a different state (transition eh5):

ApplyCharging

FurnishChargingInformation

ResetTimer

SendChargingInformation

The following operations can be received from the SCF and processed by the Handed-off SSF, causing a state transition to Waiting For End Of User Interaction state (transition eh3):

ConnectToResource

If the **ReleaseCall** operation is received from the SCF, the Handed-off SSF FSM shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

– if ApplyChargingReport operation is not requested, the Handed-off SSF FSM transits to the Idle state (transition eh2);

– if ApplyChargingReport operation has been requested, the Handed-off SSF sends ApplyChargingReport to SCF and then the Handed-off SSF FSM transits to the Idle state (transition eh2).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

Note that multiple Hand-off procedures are not covered by this Recommendation.

11.7.3 State hc: Waiting For End Of User Interaction

The FSM for Handed-off SSF enters this state from the Waiting For Instructions state (transition eh3) on the reception of one of the following operations:

ConnectToResource

During this state, the following events can occur:

– One of the following valid SCF-SRF operation for relaying is received and is correct, the operation is transferred to the SRF for execution:

Cancel(invokeID)

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptClose

ScriptInformation

ScriptRun

The FSM for Handed-off SSF remains in the Waiting For End Of User Interaction state (transition eh6).

– One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF:

SpecializedResourceReport

ReturnResult from PromptAndCollectUserInformation

ReturnResult from PromptAndReceiveMessage

ScriptEvent

The SSF for Handed-off SSF remains in the Waiting For End Of User Interaction state (transition eh6).

– When the SRF indicates to the SSF the end of user interaction by initiating disconnection the FSM for Handed-off SSF returns to the Waiting For Instructions state (transition eh4).

– The application timer T_{SSF} expires: the FSM for Handed-off SSF moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted. (transition eh7).

– An operation is received from the SCF: The FSM for Handed-off acts according to the operation received as described below.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition eh6):

ResetTimer

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the Handed-off SSF in this state, causing a transition to the Waiting For Instructions state (transition eh4). This procedure is only valid if a ConnectToResource was previously processed to cause a transition into the Waiting For End Of User Interaction state.

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

11.8 User Service Interaction USI FSM

The SSF_USI FSM illustrates state transitions for requesting and cancelling the monitoring of the reception of UTSI operations and the sending of STUI operations. This FSM has two states which are "Monitoring UTSI" and "Idle" (see Figure 11-8):

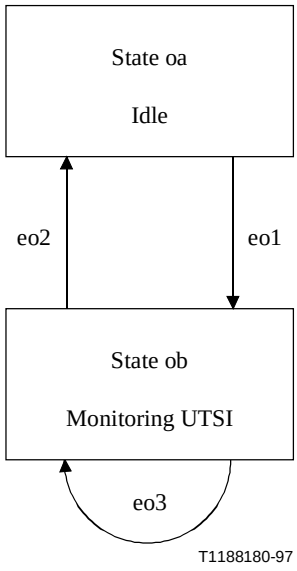


Figure 11-8/Q.1228 – FSM for SSF_USI

USI User Service Information

UTSI User to Service Information

STUI Service to User Information

The SSF_USI FSM transitions are defined in the following way:

eo1 The SCF requests the SSF to monitor the receipt of an UTSI Information Element for a given *USIServiceIndicator* value by sending a RequestReportUTSI operation for a particular party indicated by the leg identifier by setting the "USIMonitorMode" value to "monitoringActive".

eo2 The SCF requests the SSF to stop monitoring the reception of an UTSI Information Element for a given *USIServiceIndicator* value by sending a RequestReportUTSI operation for a particular party indicated by the leg identifier by setting the "USIMonitorMode" value to "monitoringInactive".

eo3 The SCF either sends an STUI Information Element by means of a SendSTUI operation to the user indicated by the leg identifier for a given *USIServiceIndicator* value; or receives by means of a ReportUTSI operation.

With the same operation, the SCF requests the SSF to monitor or to stop monitoring the receipt of an UTSI Information Elements with a given *USIServiceIndicator* value.

NOTE – As an SCF controls or monitors the call, the SSF FSM is in any state except "Idle"; but the OCCRUI mechanism does not cause any transition in the SSF FSM. When the FSM for the CallSegment returns to the Idle state then also the FSM for SSF_USI returns to Idle. SCF may send the operation SendSTUI in the state Idle of the FSM for SSF_USI.

12 SCF application entity procedures

12.1 General

This clause provides the definition of the SCF Application Entity (AE) procedures related to the SCF-SSF/SRF/SDF interfaces. The procedures are based on the use of Signalling System No. 7 (SS No. 7); other signalling systems can also be used.

In addition, other capabilities may be supported in an implementation-dependent manner in the SCP, AD or SN.

The AE, following the architecture defined in Recommendations Q.700, Q.771, and Q.1400, includes Transaction Capabilities Application Part (TCAP) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF/MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

The procedure may equally be used with other message-based signalling systems supporting the Application Layer structures defined. By no means is this text intended to dictate any limitations to Service Logic Programs (SLPs).

In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using the TCAP service, the statements and rules contained in clauses 17 and 18 shall be followed.

12.2 Model and interfaces

The functional model of the AE-SCF is shown in Figure 12-1; the ASEs interface with supporting protocol layers to communicate with the SSF, SRF and SDF, and interface to the service logic programs and maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 12-1.

NOTE – The SCF FSM includes several Finite State Machines.

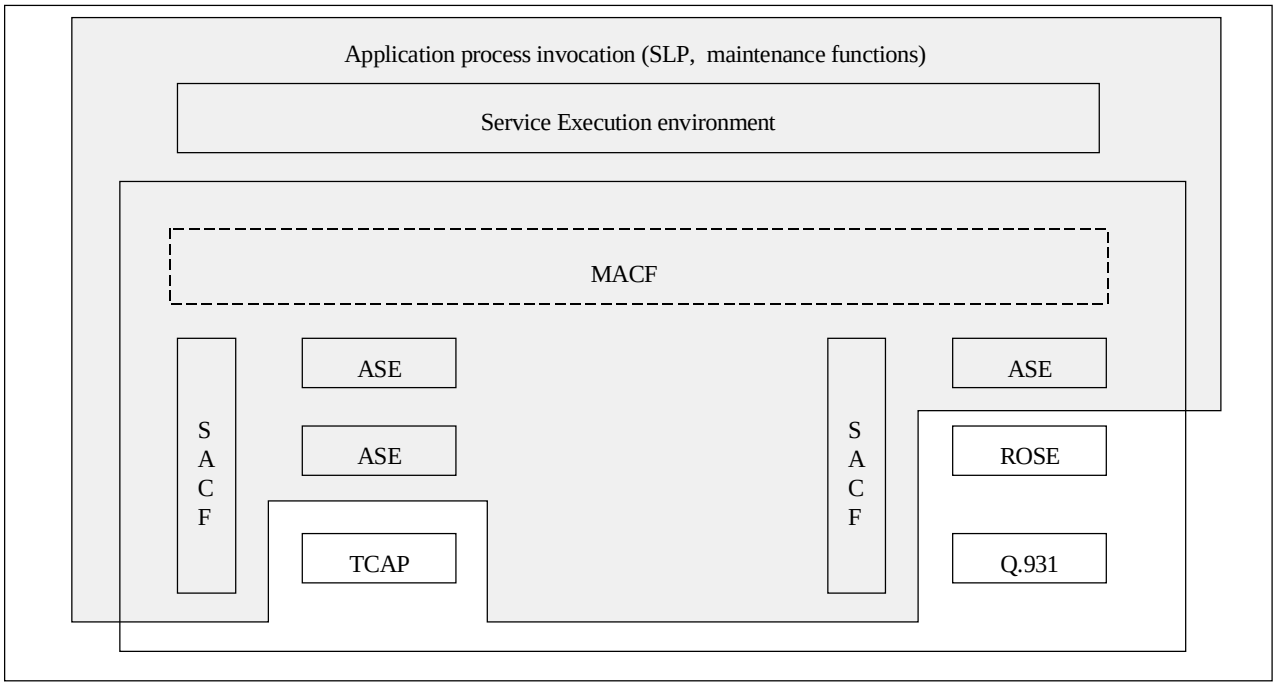


Figure 12-1/Q.1228 – Functional model of SCF AE

The interfaces shown in Figure 12-1 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of INAP are defined earlier in this Recommendation.

12.3 Relationship between the SCF FSM and the SLPs/maintenance functions

The primitive interface between the SCF FSM and the SLPs/maintenance functions is an internal interface and is not a subject for standardization.

In the SCSM instance, there are four kinds of FSMs: SSF/SRF, SDF, SCF and CUSF related states.

SSF/SRF related states consist of FSMs for SSF/SRF, for CSA, for Call Segment for Specialized Resource, for Handed-off SSF, and for Assisting SSF.

The following text systematically describes the procedural aspects of the interface between the SCF and other functional entities, with the main goal of specifying the proper order of operations rather than entities' functional capabilities. Consequently, this text describes only a subset of the SCF functional capabilities.

The relationship between the SLP and the SCF FSM may be described as follows (for both cases where a call is initiated by an end user and by IN service logic):

- If a request for IN call processing is received from the SSF, an instance of an SCF Call State Model (SCSM) is created, and the relevant SLP is invoked;
- When initiation of a call is requested from service logic, an instance of the SCSM is created.

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

The management functions related to the execution of operations received from the SCF are executed by the SCF Management Entity (SCME). Multiple requests may be executed concurrently and

asynchronously by the SCF, which explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SCF FSM objects.

The SCME is comprised of the SCME-Control and multiple instances of SCME-FSMs. The SCME Control interfaces different SCF Call State Models (SCSMs) and the Functional Entity Access Manager (FEAM). Figure 12-2 shows the SCF FSM structure.

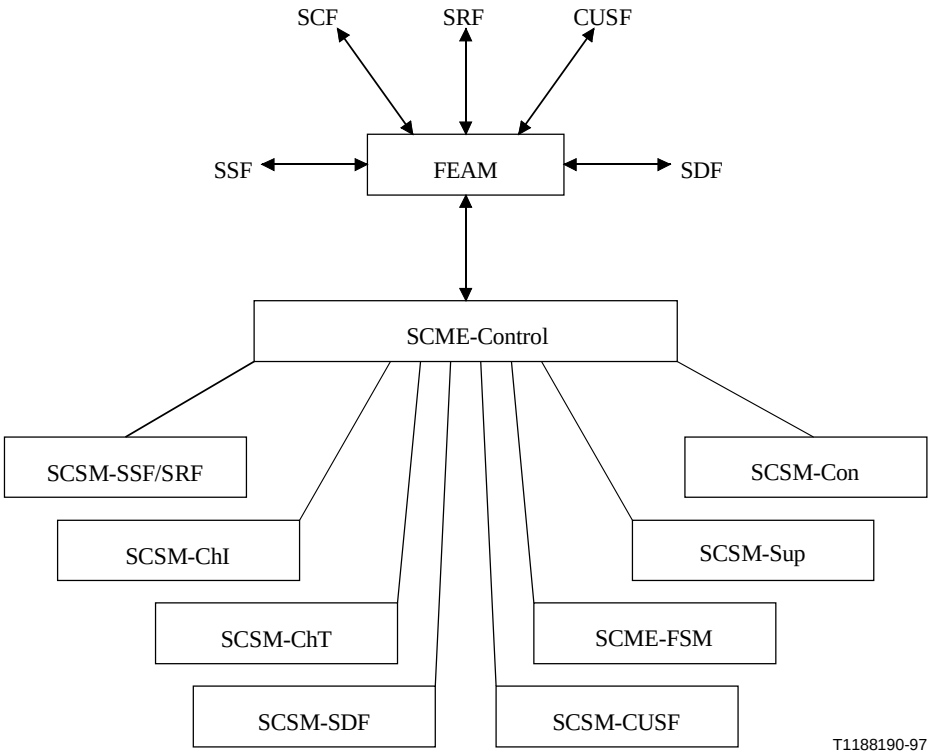


Figure 12-2/Q.1228 – SCF FSM structure

The SCME-FSM handles the interaction between the SCF and the SCF management functions.

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

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

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

The SCSM-Sup and SCSM-Con handle the interactions between SCFs, for the supporting and controlling role respectively.

The SCSM-ChI and SCSM-ChT handle the interactions between SCFs for chaining initiation and termination.

The management functions related to the execution of operations received from the SSF, SRF, SDF, CUSF or cooperating SCF are executed by the SCF Management Entity (SCME). The SCME is comprised of an SCME control and several instances of SCME-FSMs. The SCME control interfaces the different SCSMs (i.e. SCSM-SSF) and SCME-FSMs respectively as well as the Functional Entity Access Manager (FEAM).

The SCME Control maintains the associations with the SSF, SRF, SDF, CUSF and cooperating SCFs on behalf of all instances of the SCF FSMs (e.g. SCSM-SSF, SCSM-ChI). These instances of the SCF FSMs occur concurrently and asynchronously as SCF related events occur, which explains

the need for a single entity that performs the task of creation, invocation and maintenance of the SCF FSMs. In particular, the SCME Control performs the following tasks:

1) interprets the input messages from other FEs and translates them into corresponding SCSM events;

2) translates the SCSM outputs into corresponding messages to other FEs;

3) performs some asynchronous (with call processing) activities (one such activity is flow control). It is the responsibility of the SCME-control to detect nodal overload and send the Overload Indication (e.g. Automatic Call Gap) to the SSF to place flow control on queries. Other such activities include non-call associated treatment due to changes in Service Filtering, Call Gapping, or Resource Monitoring status and also provision of resource status messages to the SSF;

4) supports persistent interactions between the SCF and other FEs; and

5) performs asynchronous (with call processing) activities related to management and supervisory functions in the SCF and creates an instance of a SCME-FSM. For example, the SCME provides the non-call associated treatment due to changes in Service Filtering. Therefore, the SCME-Control separates the SCSM from the Service Filtering by creating instances of SCME-FSMs for each context of related operations.

The different contexts of the SCME-FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering, this address information is given by Filtering Criteria, i.e. all ActivateServiceFiltering operations using the same address, address the same SCME-FSM handling this specific service filtering instance. For example, ActivateServiceFiltering operations providing different Filtering Criteria cause the invocation of new SCME-FSMs.

Finally, the FEAM relieves the SCME of low-level interface functions. The functions of the FEAM include:

1) establishing and maintaining interfaces to the SSF, SRF, SDF, CUSF and cooperating SCFs;

2) passing and queueing (when necessary) the messages received from the SSF, SRF, SDF, CUSF and cooperating SCF to the SDME Control;

3) formatting, queueing (when necessary), and sending the messages received from the SCME Control to the SSF, SRF, SDF, CUSF and cooperating SCF.

Note that although the SCSM includes a state and procedures concerning queue management, this type of resource management only represents one way of managing network call queues. Another alternative is to let the SSF/CCF manage call queues; however, the technical details of how the SSF/CCF performs queue management is beyond the scope of IN. As such, the RCO (see 12.4.9) and the queueing state of the SCSM (State 2.3), along with its relevant sub-states, events and procedures, are only required and applicable in the case where queue management is performed in the SCF.

Due to the unique nature of the Disconnect operation, it can be received in any state. Because of this, it is not modelled in all conditions.

12.4 Partial SCF Management Entity (SCME) State Transition Diagram

The key parts of SCF Management Entity (SCME) State Diagram are described in Figures 12-3, 12-4, 12-5, and 12-5 *bis*.

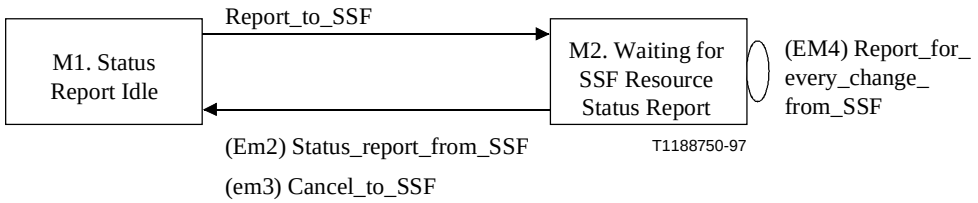


Figure 12-3/Q.1228 – The Status Report FSM in the SCME

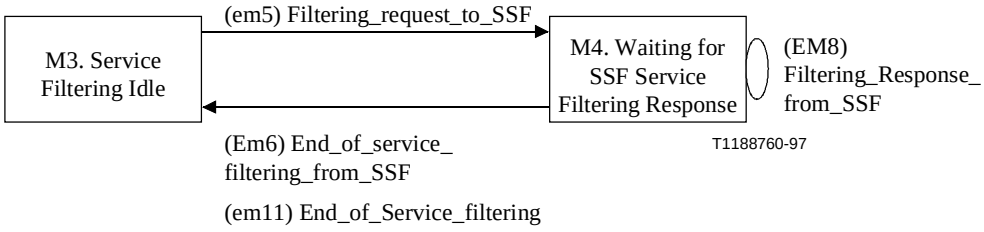


Figure 12-4/Q.1228 – The Service Filtering FSM in the SCME

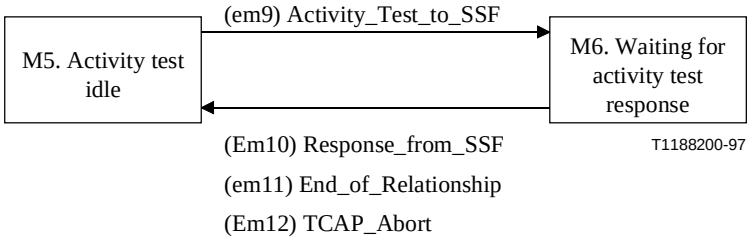


Figure 12-5/Q.1228 – The Activity Test FSM in the SCME

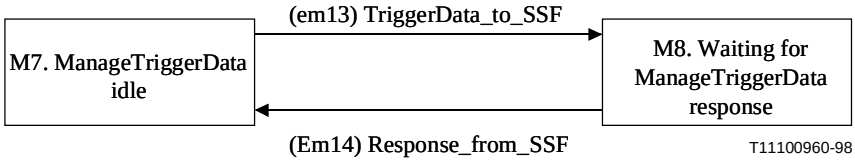


Figure 12-5 *bis*/Q.1228 – The ManageTriggerData FSM in the SCME

The SCME handles the following operations:

– **RequestCurrentStatusReport**;

– **RequestEveryStatusChangeReport**;

– **RequestFirstStatusMatchReport**;

– **CancelStatusReportRequest** (including the Resource ID previously used for **RequestFirstStatusMatchReport** or **RequestEveryStateChangeReport** operations);

– **StatusReport**;

– **ActivateServiceFiltering**;

– **ServiceFilteringResponse**;

– **CallGap**;

– **ActivityTest**; and

– **ManageTriggerData**.

Issuing the **CallGap** and the ManageTriggerData operation does not cause state transitions in the SCME. The rest of the above operations are described below.

The operations that are not listed above do not affect the state of the SCME; these operations are passed to the relevant SCSM.

12.4.1 State M1: Status report idle

The following event is considered in this state:

– (em1) Request_Status_Report_to_SSF: This is an internal event, caused by a decision to transmit one of the following operations:

  • RequestCurrentStatusReport;

  • RequestFirstStatusMatchReport;

  • RequestEveryStatusChangeReport.

This event causes a transition to state M2, waiting for SSF response status report.

12.4.2 State M2: Waiting for SSF Resource Status Report

The following events are considered in this state:

– (Em2) Status_Report_from_SSF: This is an external event, caused by the reception of the response to the RequestCurrentStatusReport or RequestFirstStatusMatchReport operation previously issued to the SSF. This event causes a transition out of this state to state M1, status report idle;

– (em3) Cancel_to_SSF: This is an internal event, caused by the service logic's need to end status monitoring of the resources in the SSF, and by transmission of CancelStatusReportRequest operation to the SSF. This event takes place only for previously issued RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operations. This event causes a transition to state M1, status report idle; and

– (Em4) Status_Report_for_Every_Change_from_SSF: This is an external event, caused by reception of the response to the RequestEveryStatusChangeReport operation previously issued to the SSF. This event does not cause a transition out of this state, so the SCME still remains in state M2, waiting for the SSF resource status report.

12.4.3 State M3: Service filtering idle

The following event is considered in this state:

– (em5) Filtering_Request_to_SSF: This is an internal event, caused by service logic's need to filter service requests to the SSF, and by transmission of the ActivateServiceFiltering operation. This event causes a transition to state M4, waiting for SSF service filtering response.

12.4.4 State M4: Waiting for SSF service filtering response

In this state, the SCF is waiting for the service filtering response from the SSF. The following events are considered in this state:

– (Em6) End_of_Service_Filtering_Response_from_SSF: This is an external event, caused by reception of the response at the end of the service filtering duration to the request service filtering previously issued to the SSF. This event causes a transition out of this state to state M3, service filtering idle;

– (em7) End_of_Service_Filtering: This is an internal event, caused by the expiration of service filtering duration timer in the SCF. This event causes a transition to state M3, service filtering idle;

– (Em8) Filtering_Response_from_SSF: This is an external event, caused by reception of the response to the request service filtering operation previously issued to the SSF. This event does not cause a transition out of this state, and the SCME remains in state M4, waiting for SSF service filtering response.

When service filtering is active, another service filtering operation could be sent to the SSF that has the same filtering criteria; this second "filter" replaces the first one.

12.4.5 State M5: Activity test idle

The following event is considered in this state:

– (em9) Activity_test_to_SSF: This is an internal event, caused by the expiration of activity test timer in the SCF, and by transmission of the ActivityTest operation. This event causes a transition to state M6, waiting for activity test response.

12.4.6 State M6: Waiting for activity test response

In this state, the SCF is waiting for the activity test response from the SSF. The following events are considered in this state:

– (Em10) Activity_Test_Response_from_SSF: This is an external event, caused by reception of the response to the activity test previously issued to the SSF. This event causes a transition out of this state to state M5, activity test idle;

– (em11) End_of_Relationship: This is an internal event, caused by the expiration of ActivityTest operation timer in the SCF. This event causes a transition to state M5, activity test idle;

– (Em12) TCAP_Abort: This is an external event, caused by reception of a P-Abort from TCAP in response to the ActivityTest operation previously issued to the SSF. This event causes a transition to state M5, activity test idle.

12.4.7 State M7: ManageTriggerData idle

The following event is considered in this state:

– (em13) TriggerData_to_SSF: This is an internal event, caused by transmission of the ManageTriggerData operation. This event causes a transition to state M8, waiting for ManageTriggerData response.

12.4.8 State M8: Waiting for ManageTriggerData activity test response

In this state, the SCF is waiting for the ManageTriggerData response from the SSF. The following event is considered in this state:

– (Em14) Response_from_SSF: This is an external event, caused by reception of the response to the ManageTriggerData previously issued to the SSF. This event causes a transition out of this state to state M7, ManageTriggerData idle.

12.4.9 The Resource Control Object

The Resource Control Object (RCO) is part of the SCF management entity that controls data relevant to resource information.

The RCO consists of:

1) a data structure that (by definition) resides in the SDF and can be accessed only via the RCO's methods; and

2) the RCO methods.

For purposes of this Recommendation, no implementation constraints are placed on the structure. The only requirement to the structure is that, for each supported resource, it:

1) stores the resource's status (e.g. busy or idle); and

2) maintains the queue of SCSMs that are waiting for this resource. For continuous monitoring, the RCO maintains its knowledge of the status of the resources through use of the request every status change report operation.

The following three methods are defined for the RCO:

1) Get_Resource: This method is used to obtain the address of an idle line on behalf of an SCSM. If the resource is busy, the SCSM is queued for it;

2) Free_Resource: This method is used when a disconnect notification from the SSF is received. The method either advances the queue (if it is not empty) or marks the resource free (otherwise); and

3) Cancel: This method is used when either the queueing timer has expired or the call has been abandoned.

12.5 The SCF Call State Model (SCSM)

In the SCSM, there are four kinds of FSMs: SSF/SRF, SDF, SCF and CUSF related states.

The SCSM instance transmits all events from SCME-Control to the appropriate FSM (for SSF/SRF, SDF, SCF, CUSF or for SCME). If the FSM instance does not exist yet, the SCME-Control creates it and transmits the event to it. The SSF/SRF FSM will then create instances of the FSM for CSA, for Specialized Resource, for Handed-off SSF or Assisting SSF, as required. The FSM for CSA will create instances of FSMs for Call Segments, as required.

General rules applicable to more than one state are as follows:

In every state, if there is an error in a received operation, the SLP and the maintenance functions are informed.

Generally the SCSM remains in the same state; however, different error treatment is possible in specific cases as described in clause 16. Depending on the class of the operation, the error can be reported to the SSF, SRF, or SDF (see Recommendation Q.774).

The general rules for one or a sequence of components sent in one or more TCAP messages, which may include a single operation or multiple operations, are specified in 11.5 SSF state transition diagram (they are not described here).

12.5.1 SSF/SRF-related states (SCSM-SSF/SRF)

SSF/SRF related states are contained in the FSMs for SSF/SRF interface, for CSA, for Call Segment, for specialized Resource, for Handed-off SSF and for Assisting SSF. The interactions between these FSMs are shown in Figure 12-6.

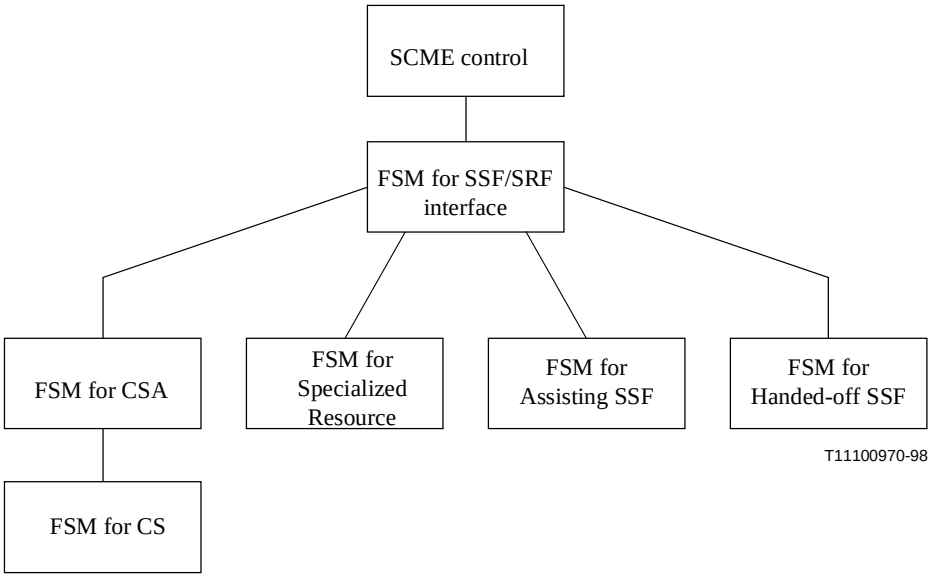


Figure 12-6/Q.1228 – FSM interactions for SCSM-SSF/SRF

The call-control-related operations relevant to the SCF-SSF-interface (except the SCME related operations) are categorized into:

1) Call-processing-related operations; and

2) Non-call-processing-related operations.

Call-processing-related operations are grouped into following two sets:

– **CollectInformation**

– **AnalyseInformation**

– **SelectFacility**

– **SelectRoute**

– **Connect**

– **Continue**

– **ContinueWithArgument**

– **Reconnect**

and:

– **InitiateCallAttempt**

– **ConnectToResource**

– **DisconnectForwardConnection**

– **DisconnectForwardConnectionWithArgument**

– **EstablishTemporaryConnection**

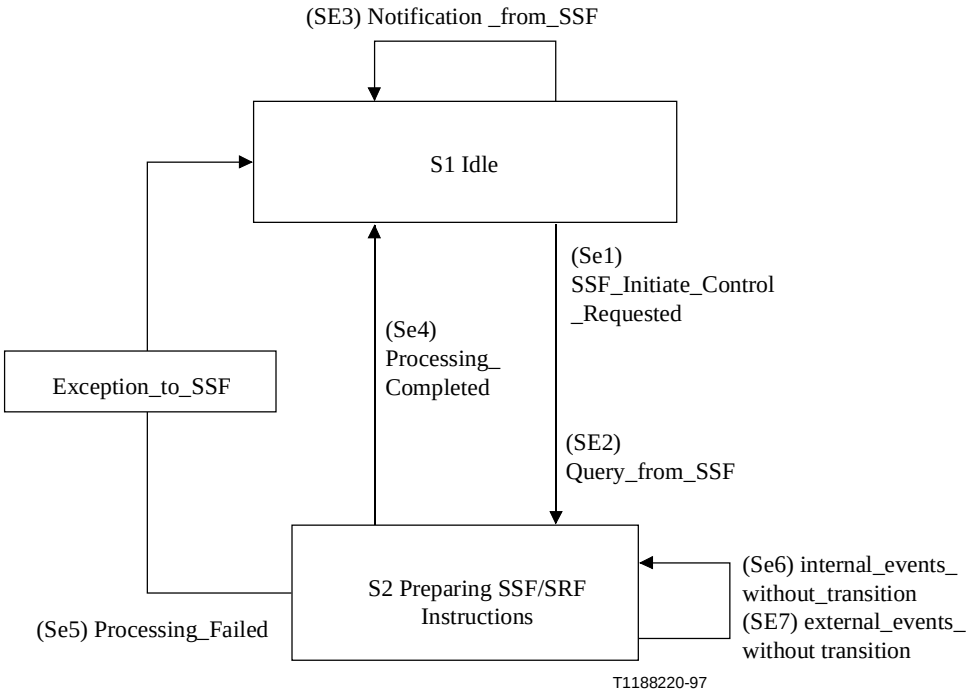
– **ReleaseCall**

For the first set of call-processing operations, the SCF may not send two operations of the same set in a series of TCAP messages or in a component sequence to the SSF, but send them only one at a time. Two operations of the first set shall be separated by at least one EDP-R message received by the SCSM. The same applies for any operation of the first set followed by ConnectToResource or EstablishTemporaryConnection.

The non-call-processing operations include the rest of the operations at the SCF-SSF interface (but not the SCME related operations). When the service logic needs to send operations in parallel, they are sent in the component sequence.

In the following, each FSM is described. The letter of "S", "I", "C", "R", "H", and "A", which are prefixed to the state numbers and the event numbers, indicates FSMs for SSF/SRF interface, for CSA, for Call Segment, for Specialized Resource, for Handed-off SSF, and for Assisting SSF, respectively.

12.5.1.1 Finite State Model for SSF/SRF interface



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-7/Q.1228 – FSM for SSF/SRF interface

Figure 12-7 shows the general State Diagram of the FSM for SSF/SRF interface as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses. The state of Preparing SSF/SRF Instructions has internal sub-FSMs composed of the sub-states.

The FSM for SSF/SRF interface has an application timer, $T_{ASSIST/HAND-OFF}$, whose purpose is to prevent excessive assist/hand-off suspension time. The FSM for SSF/SRF interface sets the timer $T_{ASSIST/HAND-OFF}$ when the SCSM sends the **EstablishTemporaryConnection** or **SelectRoute/Connect** operation with a correlation ID. This timer is stopped when the FSM for SSF/SRF interface receives the **AssistRequestInstructions** operation from the assisting/handed-off SSF or assisting SRF. On expiration of $T_{ASSIST/HAND-OFF}$, the FSM for SSF/SRF interface informs SLPI and the maintenance functions, and the FSM for SSF/SRF interface remains in the "**Preparing SSF/SRF Instructions"** state.

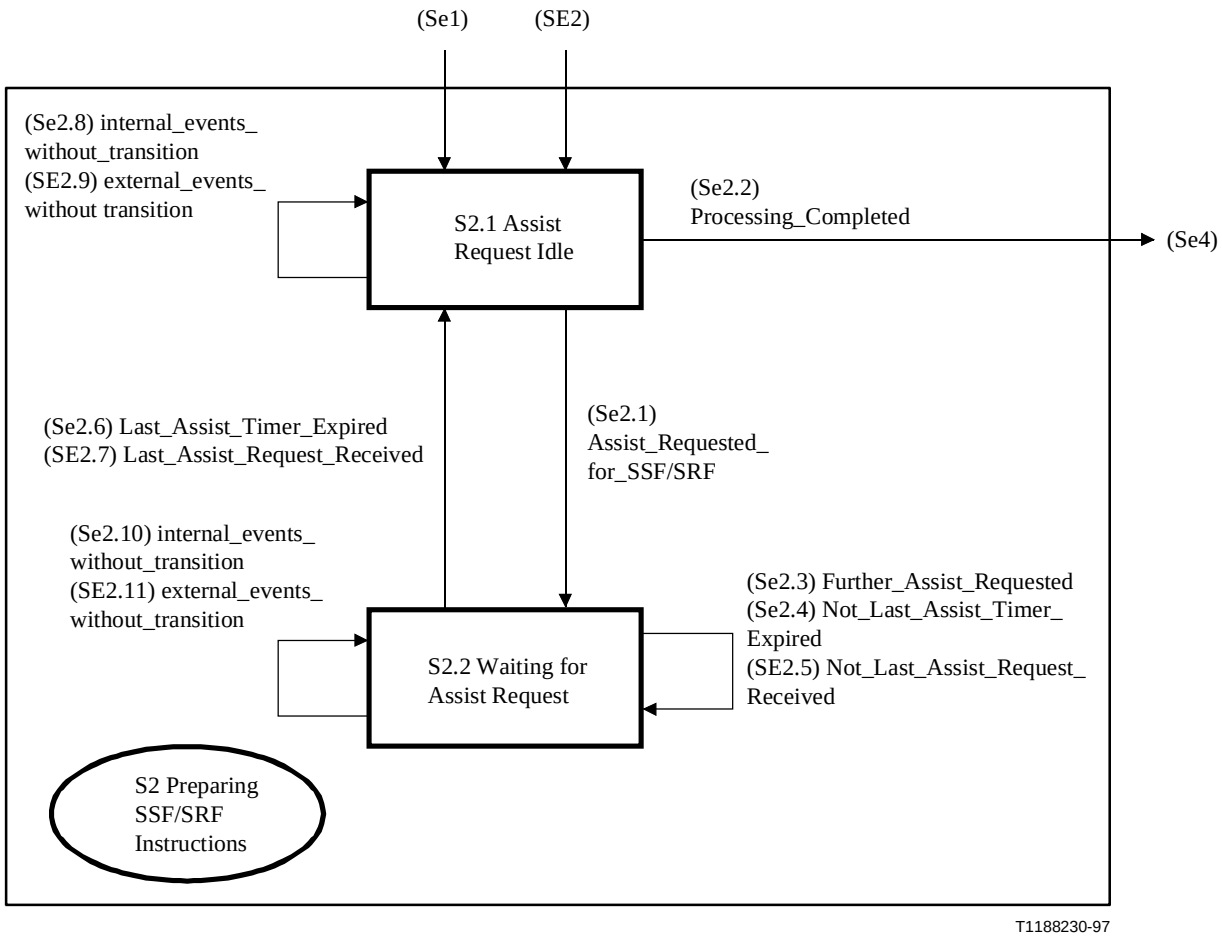
12.5.1.1.1 State S1: Idle

The following events are considered in this state:

- (Se1) SSF_Initiate_Control_Requested: This is an internal event caused by the service logic's need to have a new control relationship with SSF. The FSM for CSA requests to transmit the **InitiateCallAttempt** operation to the SSF. This event causes a transition to the state S2, **Preparing SSF/SRF Instructions.**
- (SE2) Query_from_SSF: This is an external event caused by a reception of one of the following operations:
  - **InitialDP** (for TDP-R); and
  - **DP-specific operations** (for TDP-R).

  This event causes a transition to the state S2, **Preparing SSF/SRF Instructions**. And the FSM for SSF/SRF interface creates a new FSM instance for CSA, and transmits this event to the FSM.
- (SE3) Notification_from_SSF: This is an external event caused by a reception of one of the following operations:
  - **InitialDP** (for TDP-N); and
  - **DP-specific operations** (for TDP-N).

  This event causes a transition to the same state. In this event, the FSM for SSF/SRF interface is released after the validity of the received operation is confirmed.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-8/Q.1228 – Partial expansion of the state S2 FSM

12.5.1.1.2 State S2: Preparing SSF/SRF Instructions

The following events are considered in this state:

- (Se4) Processing_Completed: This is an internal event caused by the end of service. In this case, the SCF has completed the processing for SSF and SRF. This event causes a transition to the state S1, **Idle**.
- (Se5) Processing_Failed: This (internal) event causes an appropriate exception processing and a transition back to the state S1, **Idle**.

  NOTE – Here and further in this Recommandation, the exception processing is not defined. However, it is assumed that it must include releasing all the involved resources and sending an appropriate response message to the SSF. This implies that all sub-states handle event Se5, but it is not modelled.
- (Se6) internal_events_without_transition: This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation to the corresponding FE. In this case, associate FSMs still exist. This event causes a transition to the same state.

- (SE7) external_events_without transition: This is an external event caused by receiving an event from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, associate FSMs still exist. This event causes a transition to the same state.

In this state, any events received by the FSM for SSF/SRF interface which are relevant to associate FSMs are sent to those FSMs.

In this state, when all associate FSMs are released and there are no pending application timers, $T_{ASSIST/HAND-OFF}$, the FSM instance for SSF/SRF interface transits to the state 1, **Idle** and is released. However, when all associate FSMs are released and there is a pending application timer, $T_{ASSIST/HAND-OFF}$, the FSM instance for SSF/SRF interface remains in this state.

To further describe the procedures relevant to this state, this state is divided into two sub-states, which are described in the following two subclauses. (This subdivision is illustrated in Figure 12-8.)

12.5.1.1.2.1 State S2.1: Assist Request Idle

The following events are considered in this state:

- (Se2.1) Assist_Requested_for_SSF/SRF: This is an internal event caused by the necessity of a new relationship with SSF or SRF for user interaction. In this case, SCSM sends one of the following operations to initiating SSF with an SRF address or an SSF address for routing:
  - **EstablishTemporaryConnection** (for Assisting SSF/SRF);
  - **Connect** (for Handed-off SSF); and
  - **SelectRoute** (for Handed-off SSF).

  This event causes a transition to the state S2.2, **Waiting for Assist Request.** The FSM for SSF/SRF interface starts the timer $T_{ASSIST/HAND-OFF}$.
- (Se2.2) Processing_Completed: This is an internal event. This event causes the transition that maps into the FSM event (Se4) for SSF/SRF interface.
- (Se2.8) internal_events_without_transition: This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation (except an **EstablishTemporaryConnection** operation and a **Connect/SelectRoute** operation for hand-off) to the corresponding FE. In this case, associate FSMs still exist. This event causes a transition to the same state.
- (SE2.9) external_events_without transition: This is an external event caused by receiving an event from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, associate FSMs still exist. This event causes a transition to the same state.

12.5.1.1.2.2 State S2.2: Waiting for Assist Request

In this state, the FSM for SSF/SRF waits for the **AssistRequestInstructions** operation from the Handed-off/Assisting SSF (SSF relay case) or from the SRF (Direct SCF-SRF case). The following events are considered in this state:

- (Se2.3) Further_Assist_Requested: This is an internal event caused by the necessity of a new relationship with SSF or SRF for user interaction. In this case, SCSM sends one of the following operations to initiating SSF with an SRF address or an SSF address for routing:
  - **EstablishTemporaryConnection** (for Assisting SSF/SRF);
  - **Connect** (for Handed-off SSF); and
  - **SelectRoute** (for Handed-off SSF).

This event causes a transition to the same state. The FSM for SSF/SRF interface starts the new timer $T_{ASSIST/HAND-OFF}$.

- (Se2.4) Not_Last_Assist_Timer_Expired: This is an internal event caused by the expiration of the timer $T_{ASSIST/HAND-OFF}$ which is one of pending timers. In this case, the FSM for SSF/SRF interface informs the SLPI, and remains in the same state.
- (SE2.5) Not_Last_Assist_Request_Received: This is an external event caused by a reception of the **AssistRequestInstructions** operation. In this case, the FSM for SSF/SRF interface stops the corresponding timer, creates a new FSM instance (FSM instance for Specialized Resource, Handed-off SSF, or Assisting SSF), and transmits the event to the new FSM instance. The FSM for SSF/SRF interface remains in the same state.
- (Se2.6) Last_Assist_Timer_Expired: This is an internal event caused by the expiration of the timer $T_{ASSIST/HAND-OFF}$ which is the last pending timer. In this case, the FSM for SSF/SRF interface informs the SLPI, and transits to the state S2.1, **Assist Request Idle**. Any other pending timers are ignored.
- (SE2.7) Last_Assist_Request_Received: This is an external event caused by a reception of the **AssistRequestInstructions** operation. In this case, the FSM for SSF/SRF interface stops the corresponding timer, creates a new FSM instance (FSM instance for Specialized Resource, Handed-off SSF, or Assisting SSF), and transmits the event to the new FSM instance. The FSM for SSF/SRF interface transits to the state S2.1, **Assist Request Idle**.
- (Se2.10) internal_events_without_transition: This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation (except an **EstablishTemporaryConnection** operation and a **Connect/SelectRoute** operation for hand-off) to the corresponding FE. In this case, any associate FSMs which are waiting for the **AssistRequestInstructions** operation still exist. This event causes a transition to the same state.
- (SE2.11) external_events_without transition: This is an external event caused by receiving an event (except an **AssistRequestInstructions** operation) from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, any associate FSMs which are waiting for the **AssistRequestInstructions** operation still exist. This event causes a transition to the same state.

12.5.1.2 Finite State Model for CSA

Figure 12-9 shows the general State Diagram of the FSM for CSA as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

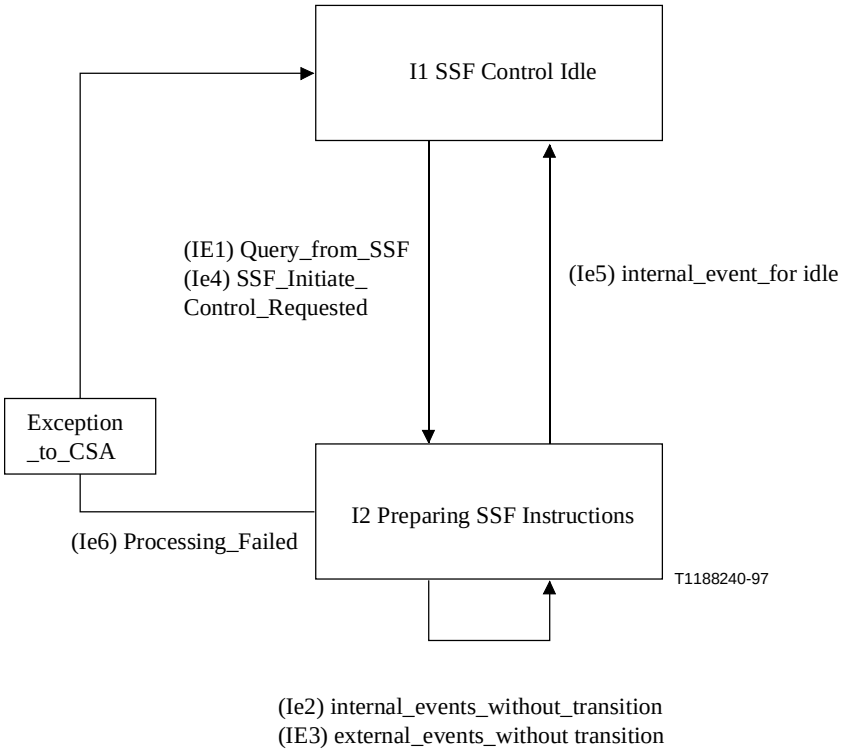


Figure 12-9/Q.1228 – FSM for CSA

The FSM for CSA receives external events from the FSM for SSF/SRF interface and either processes them directly or passes them to the relevant associate FSM for CS. It receives internal events from the SLPI or an associate FSM for CS instructing it to send operations to the FSM for SSF/SRF interface for sending to external FEs. It will also receive an internal notification of operations sent by the FSM for the SSF/SRF interface which affect the FSM for CSA.

12.5.1.2.1 State I1: SSF Control Idle

The FSM for CSA enters the SSF Control Idle state when one of the following occurs:

– when all the FSM for Call Segment instances associated with the FSM for CSA instance are released.

When the FSM for CSA enters the SSF Control Idle state, the associate FSM for SSF/SRF interface must be notified.

The following events are considered in this state:

- (IE1) Query_from_SSF: This is an external event caused by a reception of one of the following operations:
  - **InitialDP** (for TDP-R); and
  - **DP-specific operations** (for TDP-R).

  This event causes a transition to the state I2, **Preparing SSF Instructions**. The FSM for CSA creates a new FSM for Call Segment instance, and transmits this event to the FSM.
- (Ie4) SSF_Initiate_Control_Requested: This is an internal event caused by the service logic's need to have a new control relationship with SSF. This event occurs in the following cases.

– The FSM for Call Segment requests the FSM for CSA to transmit the **InitiateCallAttempt** operation to the SSF.
– The FSM for Call Segment requests the FSM for CSA to transmit the **CreateCallSegmentAssociation** operation to the SSF.

This event causes a transition to state I2, **Preparing SSF Instructions.**

12.5.1.2.2 State I2: Preparing SSF Instructions

In this state, the FSM for CSA instance handles instructions from the SCF and events which are received from the FSM for CS instances or the FSM for SSF/SRF interface.

The following events are considered in this state:

• (Ie2) internal_events_without_transition: This is an internal event caused by the following cases.
  – When the SLPI instructs the FSM for CSA instance to send the following operations to the FSM for SSF/SRF interface:
    • **FurnishChargingInformation**
    • **Cancel**(allRequests)
    • **ReleaseCall**
    • **MoveLeg**
    • **SplitLeg**

    NOTE 1 – In this case, the SSF creates a Call Segment and connects the split Leg with the Call Segment. The FSM for CSA transmits the event to the FSM for the 'source' CS instance. Furthermore, the FSM for CSA creates a new FSM for the new Call Segment instance and transmits the event to the FSM.

    • **MergeCallSegments**

    NOTE 2 – In this case, the SSF deletes the 'source' Call Segment and connects the Leg in the 'source' Call segment with the 'target' Call Segment. The FSM for CSA transmits the event to the FSM for the 'source' CS instance and releases the FSM instance. Furthermore, the FSM for CSA transmits the event to the FSM for the 'target' CS instance.

    • **RequestReportBCSMEvent**
  – When the FSM for SSF/SRF interface has sent the following operation:
    • **MoveCallSegments**
  – When the associate FSM for CS instance requests the sending of the following operations:
    • **ApplyCharging**
    • **Authorize Termination**
    • **CallInformationRequest**
    • **RequestNotificationChargingEvent**
    • **SendChargingInformation**
    • **HoldCallInNetwork**
    • **ResetTimer**
    • **Cancel**(invokeID)
    • **ConnectToResource**
    • **EstablishTemporaryConnection**

- **DisconnectForwardConnection**
- **DisconnectForwardConnectionWithArgument**
- **PlayAnnouncement**
- **PromptAndCollectUserInformation**
- **PromptAndReceiveMessage**
- **InitiateCallAttempt**
- **Connect**
- **CollectInformation**
- **AnalyseInformation**
- **ScriptInformation**
- **ScriptRun**
- **ScriptClose**
- **SelectRoute**
- **SelectFacility**
- **SendFacilityInformation**
- **Continue**
- **ContinueWithArgument**
- **Reconnect**
- **RequestReportFacilityEvent**
- **Release Call**
- **DisconnectLeg**

– When the application timer $T_{assist/hand\text{-}off}$ in the FSM for SSF/SRF interface expires.

In this case, any associate FSMs still exist. This event causes a transition to the same state.

• (IE3) external_events_without transition: This is an external event caused by receiving an event from the other FEs. The FSM for CSA will process the event and, if necessary, pass the event to the relevant associate FSMs.

– **EventReportBCSM**
– **EventReportFacility**
– **DP-Specific operations** (for EDP)
– **CallInformationReport**
– **ApplyChargingReport**
– **EntityReleased**
– **EventNotificationCharging**
– **SpecializedResourceReport**
– **ReturnResult from PromptAndReceiveMessage**
– **ScriptEvent**
– **ReturnResult for PromptAndCollectUserInformation**

In this case, any associate FSMs are existing. This event causes a transition to the same state.

• (Ie5) internal_event_for idle: This is an internal event caused by the following cases.

– When the last FSM instance for CS transits to the idle state.

In this case, any associate FSMs are not existing. This event causes a transition to the state I1, **SSF Control Idle**.

• (Ie6) Processing_Failed: This (internal) event causes an appropriate exception processing and a transition to the state I1, **SSF Control Idle**.

12.5.1.3 Finite State Model for Call Segment

Figure 12-10 shows the general State Diagram of the FSM for Call Segment as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

The following operations (the CPH operations) are of class 1 and require reception of a Return Result:

– **DisconnectLeg**

– **MergeCallSegments**

– **MoveCallSegments**

– **MoveLeg**

– **SplitLeg**

Reception of the Return Result for the operations mentioned above does not result in a state transition in the FSM for CS. Furthermore, this Return Result may be received in any state of the FSM for CS.

Reception of Return Result may support keeping an accurate CV in SCF and decide what subsequent action can be taken by the service logic. If various components (e.g. including CPH operations) are grouped into a single TC message, the return result will provide the invoke identifier. This allows the service logic to correlate the result to a previously sent CPH operation and hereby distinguish between the grouped components (operations) sent.

The FSM for CS has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, T_{SSF}, to guard the release of the call segment which is waiting for instructions from the SCF. The use of timer $T_{SCF\text{-}SSF}$ for the FSM for CS is mandatory.

The timer $T_{SCF\text{-}SSF}$ is set in the following cases:

i) when the SCF receives an **InitialDP** operation, or one of **DP-specific operations**. In this case, this timer is restart when a first request, other than **ResetTimer** operation, is sent to the SSF. On the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for CS may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF\text{-}SSF}$. On the second expiration of $T_{SCF\text{-}SSF}$, the FSM for CS informs the SLPI and the maintenance functions, and the FSM for CS transitions to the state C1, **CS Control Idle**;

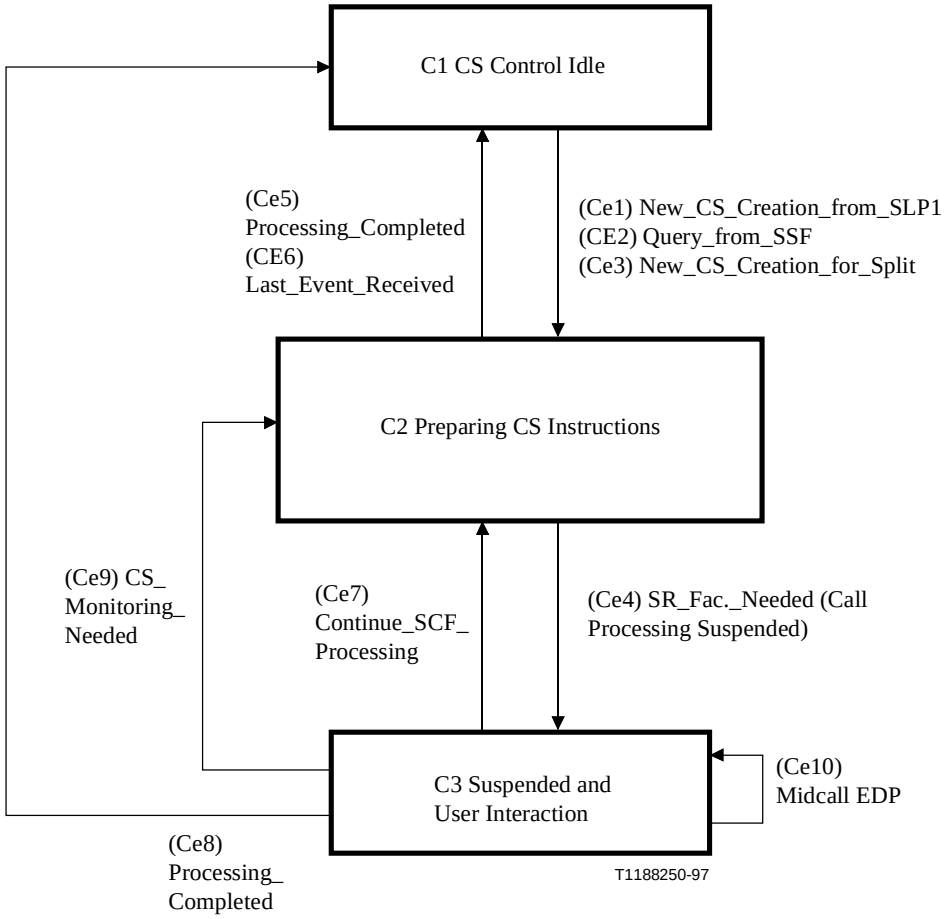
ii) when the FSM instance for CS enters the **Preparing CS Instructions** state (C2.1) under any other condition than the case i). In this case, the FSM for CS may restart T_{SSF} using the **Reset Timer** operation any number of times;

iii) when the FSM for CS enters the Queueing substate (see 12.5.1.3.2.3, State C2.3: "**Queueing**"). In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for CS may restart T_{SSF} using the **ResetTimer** operation any number of times; and

iv) when the SCF enters the **Suspended and User Interaction** state (C3). In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for CS may restart T_{SSF} using the **ResetTimer** operation any number of times.

In each of the cases, $T_{SCF-SSF}$ may have different values as defined by the application. The values of $T_{SCF-SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF-SSF}$. Whenever the value of T_{SSF} is changed by the SCF sending a **ResetTimer** operation to the SSF, the value of $T_{SCF-SSF}$ must be changed accordingly.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-10/Q.1228 – FSM for Call Segment

12.5.1.3.1 State C1: CS Control Idle

The FSM for CS must enter the **CS Control Idle** state when the associate FSM for CSA instance transits to the **SSF Control Idle** state.

When the FSM for CS enters the **CS Control Idle** state, the associate FSM for CSA must be notified.

The following events are considered in this state:

- (Ce1) New_Call_Segment_from_SLPI: This is an internal event caused by the SLPI when there is a need to send the following operation to the SSF:
  - **InitiateCallAttempt**

  This event causes a transition to the state C2, **Preparing CS Instructions**.

- (CE2) Query_from_SSF: This is an external event caused by a reception of one of the following from the SSF:
  - **InitialDP**
  - **DP-specific Request Instructions**

  This event causes a transition to the state C2, **Preparing CS Instructions**.
- (Ce3) New_Call_Segment_for_Split: This is an internal event caused by the SLPI when there is a need to send the following operation to the SSF:
  - **SplitLeg** (target CS)

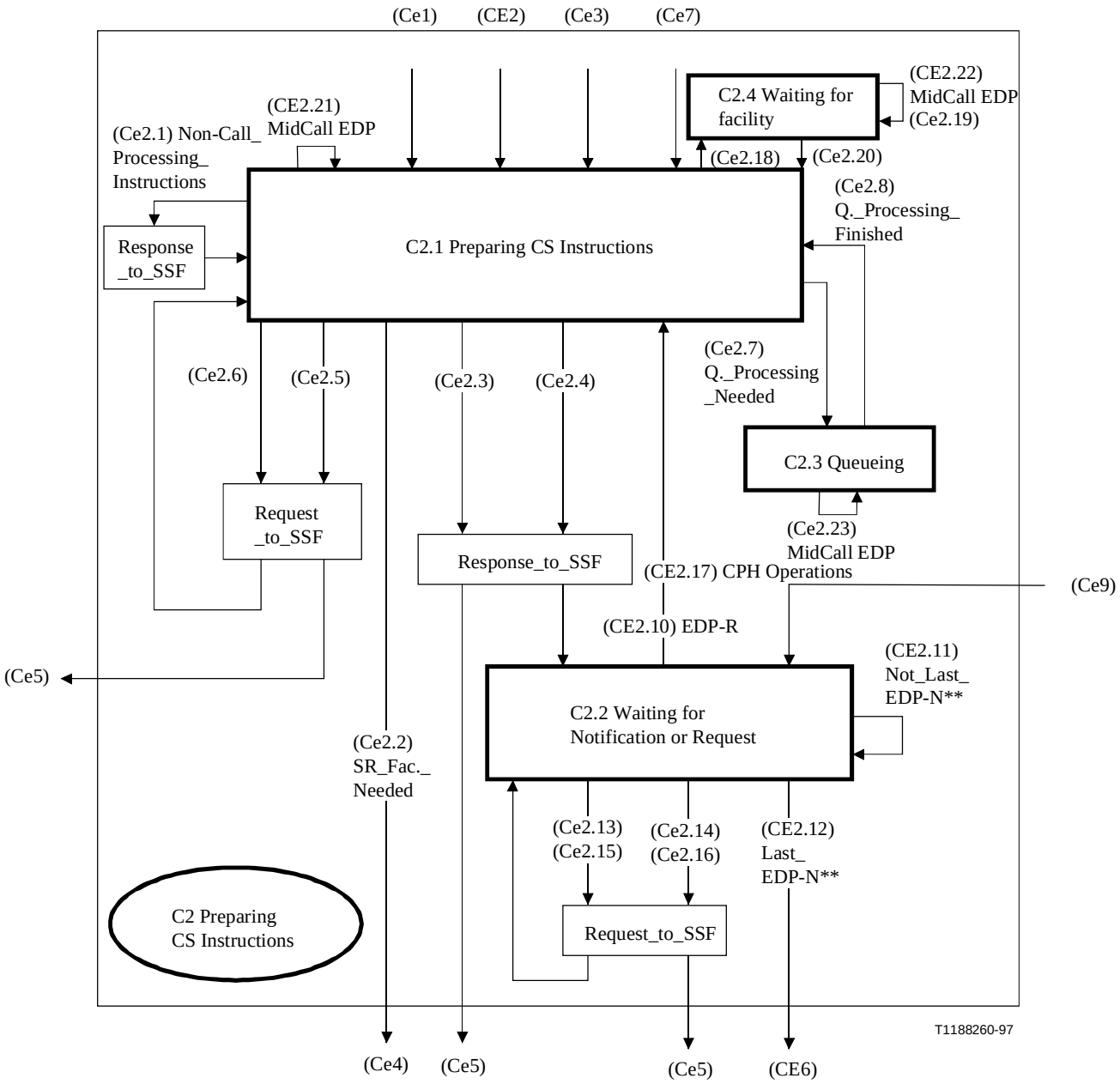
  This event causes a transition to the state C2, **Preparing CS Instructions**.

12.5.1.3.2 State C2: Preparing CS Instructions

The following events are considered in this state:

- (Ce4) SR_Facilities_Needed (Call Processing Suspended): This is an (internal) event caused by the service logic's need for additional information from the call party; hence is the necessity to set up a connection between the call party and the SRF. This event causes a transition to state C3, **Suspended and User Interaction**.
- (Ce5) Processing_Completed: This is an internal event, caused by the end of processing for the CS. This event causes a transition to the state C1, **CS Control Idle**.
- (CE6) Last_Event_Received: This is an external event caused by a reception of a last event from the SSF.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in Figure 12-11.



(CE2.3): Call_Processing_Instruction_Ready (Monitoring* not Required)
(Ce2.4): Call_Processing_Instruction_Ready (Monitoring* Required)
(Ce2.5): CS_or_Leg_Control_Last_Instruction
(Ce2.6): CS_or_Leg_Control_Continuing_Instruction)
(Ce2.13): Notification_or_Request_Continuing_Instruction
(Ce2.14): Monitoring_Cancel_Instruction
(Ce2.15): Release_Call_Instruction (Call Information Report or Apply Charging Report has been requested)
(Ce2.16): Release_Call_Instruction (Neither Call Information Report nor Apply Charging Report has been requested)
(Ce2.17): CPH Operations
(Ce2.18): First Publicity Events
(Ce2.19): Subsequent Facility Event
(Ce.2.20): Last Facility Event

Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-11/Q.1228 – Partial expansion of the state C2 FSM for CS

12.5.1.3.2.1 State C2.1: Preparing CS instructions

The following events are considered in this state:

- (Ce2.1) Non-Call_Processing_Instructions: This is an internal event caused by the following cases:

  i) When the service logic needs to send an operation such as following to the SSF:
  - **ApplyCharging**;
  - **CallInformationRequest**;
  - **SendFacilityInformation**;
  - **RequestNotificationChargingEvent**; and
  - **SendChargingInformation**.

  ii) When the following operations for sending to the SSF are notified from the FSM for CSA:
  - **Cancel**(allRequests);
  - **RequestReportBCSMEvent**.

  iii) When the application timer $T_{SCF-SSF}$ expires. In this case, the FSM for CS sends a **ResetTimer** operation to the corresponding CS FSM in the SSF.

  This event causes a transition back to state C2.1, **Preparing CS Instructions**.

- (Ce2.2) SR_Facilities_Needed: This is an internal event, caused by the service logic when there is a need to use the SRF. This event is mapped as the FSM event (Ce4).

- (Ce2.3) Call_Processing_Instruction_Ready (monitoring not required): This is an internal event caused by the service logic when the final call-processing-related operation is ready and there is neither an armed EDP nor an outstanding **CallInformationReport** or **ApplyChargingReport** operation. It causes one of the following operations to be issued to the SSF:
  - **AnalyseInformation**;
  - **AuthorizeTermination**;
  - **CollectInformation**;
  - **Connect**;
  - **Continue**;

    (Only applicable for a single CS with no more than two legs. Use of this operation is not valid in a multicall segment CSA.)
  - **ContinueWithArgument**;
  - **Reconnect**;
  - **ReleaseCall**;
  - **SelectFacility**; and
  - **SelectRoute**.

  In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:
  - **Cancel**(allRequests);
  - **RequestReportBCSMEvent** (to disarm all the armed EDPs);
  - **RequestNotificationChargingEvent**; and (to set mode transparent to stop monitoring for charging events); and

– **SendChargingInformation**.

This event is mapped as the FSM event (Ce5).

- (Ce2.4) Call_Processing_Instruction_Ready (monitoring required): This is an internal event caused by the service logic when a call-processing-related operation is ready and the monitoring of the call is required (e.g. an EDP is armed, or there is an outstanding **CallInformationReport** or **ApplyChargingReport**). It causes one of the following operations to be issued to the SSF:
  – **AnalyseInformation**;
  – **AuthorizeTermination**;
  – **CollectInformation**;
  – **Connect**;
  – **Continue**;

    (Only applicable for a single CS with no more than two legs. Use of this operation is not valid in a multicall segment CSA.)
  – **ContinueWithArgument**;
  – **Reconnect**;
  – **ReleaseCall**;
  – **SelectFacility**; and
  – **SelectRoute**.

  In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:
  – **ApplyCharging**;
  – **CallInformationRequest**;
  – **RequestReportBCSMEvent**;
  – **RequestNotificationChargingEvent**; and
  – **SendChargingInformation**.

  This event causes a transition to the state C2.2, **Waiting for Notification or Request**.
- (Ce2.5) CS_or_Leg_Control_Last_Instruction: This event is an internal event caused by the following cases:
  i) When the SLPI needs to send an operation such as the following to the SSF:
     – **DisconnectLeg** (for last leg)
  ii) When the following operations have been sent to the SSF by an associate FSM for CSA instance:
     – **MergeCallSegments** (for 'source' CS)

  This event is mapped as the FSM event (Ce5).
- (Ce2.6) CS_or_Leg_Control_Continuing_Instruction: This event is an internal event caused by the following cases.
  i) When the SLPI needs to send an operation such as the following to the SSF:
     – **DisconnectLeg** (for not last leg)

ii) When the following operations have been sent to the SSF by an associate FSM for CSA instance:

- **MergeCallSegments** (for "target" CS)
- **SplitLeg** (for "source" CS)
- **MoveLeg**

This event causes a transition back to the same state.

- (Ce2.7) Queueing_Processing_Needed: This is an internal event caused by the service logic when queueing of the call is required. This event causes a transition into state C2.3, **Queueing**.
- (CE2.21) MidCallEDP: This event is an external event caused by a reception of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".

12.5.1.3.2.2 State C2.2: Waiting for Notification or Request

The following events are considered in this state:

- (CE2.10) EDP-R: This is an external event, caused by a reception of one of the following operations:
  - **EventReportBCSM** (for EDP_R);
  - **AnalysedInformation** (for EDP_R);
  - **CollectedInformation** (for EDP_R);
  - **FacilitySelectedandAvailable** (for EDP-R);
  - **OAbandon** (for EDP_R);
  - **OAnswer** (for EDP_R);
  - **OCalledPartyBusy** (for EDP_R);
  - **ODisconnect** (for EDP_R);
  - **OMidCall** (for EDP_R);
  - **ONoAnswer** (for EDP_R);
  - **OriginationAttemptAuthorized** (for EDP-R);
  - **OSuspended** (for EDP_R);
  - **RouteSelectFailure** (for EDP_R);
  - **TAnswer** (for EDP_R);
  - **TBusy** (for EDP_R);
  - **TDisconnect** (for EDP_R);
  - **TermAttemptAuthorized** (for EDP_R);
  - **TMidCall** (for EDP_R);
  - **TNoAnswer** (for EDP_R); and
  - **TSuspended** (for EDP_R).

  This event causes a transition to the state C2.1, **Preparing CS Instructions**.
- (CE2.11) Not_Last_EDP-N: This is an external event, caused by a reception of one of the following operations:
  - **ApplyChargingReport**;
  - **CallInformationReport**;

– **EventReportBCSM** (for EDP_N);
– **EventNotificationCharging**;
– **AnalysedInformation** (for EDP_N);
– **CollectedInformation** (for EDP_N);
– **FacilitySelectedandAvailable** (for EDP-N);
– **OAbandon** (for EDP_N);
– **OAnswer** (for EDP_N);
– **OCalledPartyBusy** (for EDP_N);
– **ODisconnect** (for EDP_N);
– **OMidCall** (for EDP_N);
– **ONoAnswer** (for EDP_N);
– **OriginationAttemptAuthorized** (for EDP-N);
– **OSuspended** (for EDP_N);
– **RouteSelectFailure** (for EDP_N);
– **TAnswer** (for EDP_N);
– **TBusy** (for EDP_N);
– **TDisconnect** (for EDP_N);
– **TermAttemptAuthorized** (for EDP_N);
– **TMidCall** (for EDP_N);
– **TNoAnswer** (for EDP_N); and
– **TSuspended** (for EDP_N).

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. This event causes a transition back to the state C2.2, **Waiting for Notification or Request**.

Refer to the meaning of "last EDP-N" which is described in the event (CE2.12).

• (CE2.12) Last_EDP-N: This is an external event, caused by a reception of one of the following operations:
– **ApplyChargingReport**;
– **CallInformationReport**;
– **EntityReleased**;
– **EventReportBCSM** (for EDP_N);
– **AnalysedInformation** (for EDP_N);
– **CollectedInformation** (for EDP_N);
– **FacilitySelectedandAvailable** (for EDP-N);
– **OAbandon** (for EDP_N);
– **OAnswer** (for EDP_N);
– **OCalledPartyBusy** (for EDP_N);
– **ODisconnect** (for EDP_N);
– **OMidCall** (for EDP_N);
– **ONoAnswer** (for EDP_N);

– **OriginationAttemptAuthorized** (for EDP-N);
– **OSuspended** (for EDP_N);
– **RouteSelectFailure** (for EDP_N);
– **TAnswer** (for EDP_N);
– **TBusy** (for EDP_N);
– **TDisconnect** (for EDP_N);
– **TermAttemptAuthorized** (for EDP_N);
– **TMidCall** (for EDP_N);
– **TNoAnswer** (for EDP_N); and
– **TSuspended** (for EDP_N).

In this case, there is no outstanding armed EDP and neither pending **CallInformationReport** nor pending **ApplyChargingReport**. This event is mapped as the FSM event (CE6).

NOTE – The 'last EDP-N' means that there are no other EDPs which may be encountered when an EDP-N was detected. Some of the EDPs are automatically disarmed if another EDP is encountered. The EDPs which are automatically disarmed depend on which EDP is encountered. An example is the case of the EDPs O_Answer, O_No_Answer, RouteSelectFailure or O_Called_Party_Busy. If any of these EDPs are encountered, all the other EDPs of this list are automatically disarmed.

- (Ce2.13) Notification_or_Request_Continuing_Instruction: This event is an internal event caused by the following cases:

i) When the SLPI needs to send an operation such as the following to the SSF:
   – **ApplyCharging**;
   – **FurnishChargingInformation**;
   – **RequestNotificationChargingEvent**; and
   – **SendChargingInformation**.

ii) When the following operations have been sent to the SSF by the FSM for CSA:
   – **RequestReportBCSMEvent** (for the purpose of which:
      a) one or more of EDPs will be armed;
      b) a part of armed EDPs will be disarmed; or
      c) all of armed EDPs will be disarmed when there are other pending requests);

This event causes a transition back to the state C2.2, **Waiting for Notification or Request**.

- (Ce2.14) Monitoring_Cancel_Instruction: This is an internal event caused when the associate FSM for CSA instance has sent one of the following operations to the SSF:
   – **RequestReportBCSMEvent** (for the purpose of which all of armed EDPs will be disarmed when there are no more other pending requests); and
   – **Cancel**(allRequests).

This event is mapped as the FSM event (Ce5).

- (Ce2.15) Release_Call_Instruction (Call Information Report or Apply Charging Report has been requested): This is an internal event caused when the associate FSM for CSA instance has sent the following operation to the SSF:
   – **ReleaseCall** (when there is outstanding **CallInformationReport** or **ApplyChargingReport**).

This event causes a transition back to the state C2.2, **Waiting for Notification or Request**.

- (Ce2.16) Release_Call_Instruction (neither Call Information Report nor Apply Charging Report has been requested): This is an internal event caused when the associate FSM for CSA instance has sent the following operation to the SSF:
  - **ReleaseCall** (when there is no outstanding **CallInformationReport** or **ApplyChargingReport**).

  This event is mapped as the FSM event (Ce5).
- (Ce2.17) CPH Operation Instruction: This is an internal event caused by the following cases:
  i) When the SLPI needs to send an operation such as the following to the SSF:
     - **DisconnectLeg** (not for last leg)

  ii) When the following operations have been sent to the SSF by the associate FSM for CSA instance:
     - **MergeCallSegments** (for 'target' CS)
     - **MoveLeg**
     - **SplitLeg** (for 'source' CS)

  This event causes a transition back to the state C2.1, **Preparing CS Instructions**.

12.5.1.3.2.3 State C2.3: Queueing

When the SCF is processing the query from the SSF/CCF, it may find that the resource to which the call shall be routed is unavailable. One possible reason causing the resource to be unavailable is the "busy" condition.

NOTE – The manner in which the status of the resources is maintained is described in 12.4.7.

Such a resource may be an individual line or trunk or a customer-defined group of lines or trunks. In the latter case, the word "busy" means that all lines or trunks in the group are occupied; and the word "idle" means that at least one line or trunk in the group is idle.

If the resource is busy, the SCF may put the call on queue and resume it later when the resource is idle. The following operations can be sent in this state:

- **HoldCallInNetwork**;
- **ApplyCharging**;
- **CallInformationRequest**;
- **RequestReportBCSMEvent**;
- **RequestNotificationChargingEvent**;
- **ResetTimer**; and
- **SendChargingInformation**.

The following events are considered in this state:

- (Ce2.8) Queueing_Processing_Finished: This is an internal event caused by the SLP when it is ready to prepare the call-related operation for sending to the SSF. This event causes a transition to State C2.1, **Preparing CS Instructions**.

  This state further expands into an FSM, which is depicted in Figure 12-12.

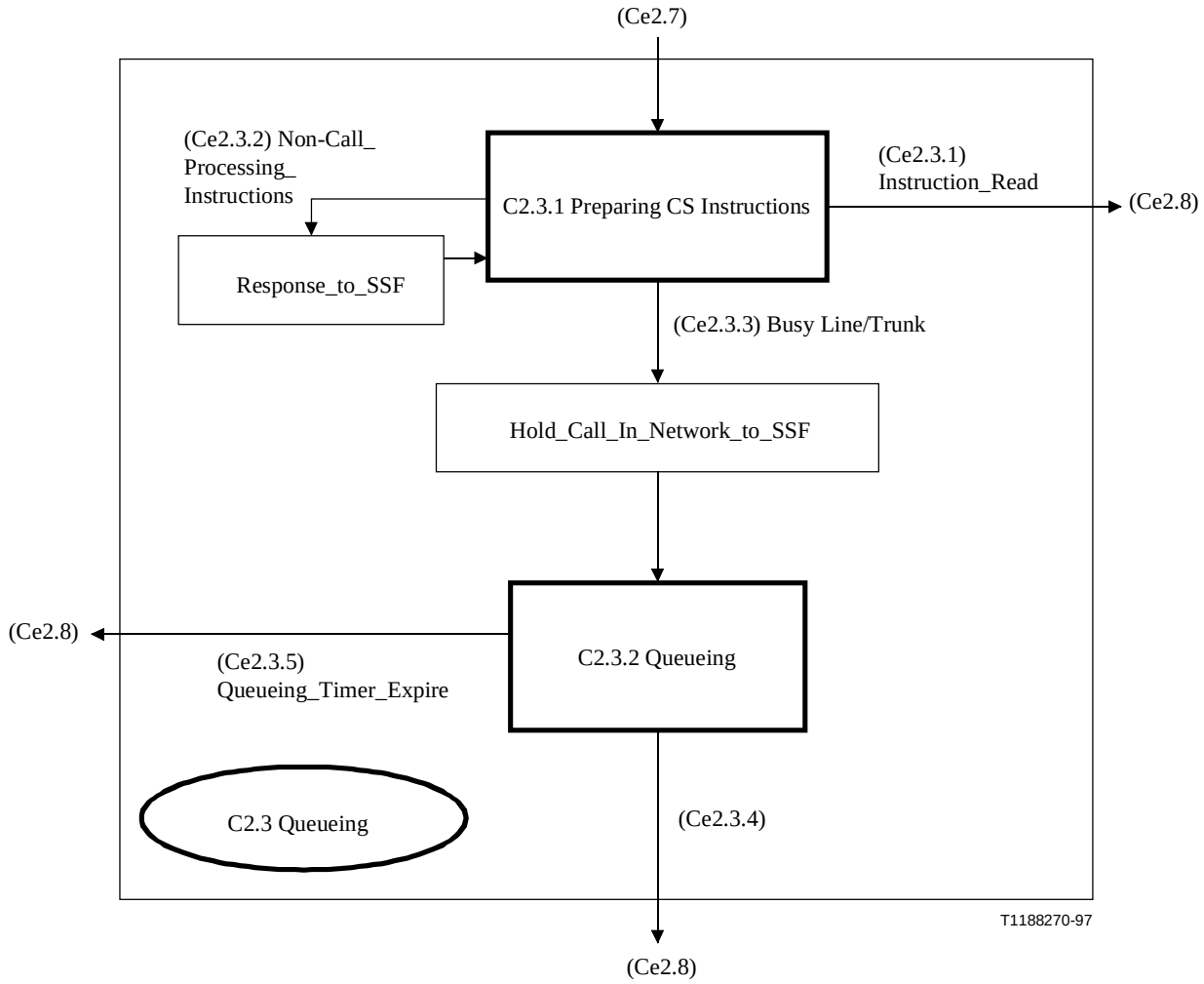
  This FSM does not explicitly describe all possible combinations of resource monitoring functions used for queueing. The following possibilities may be used in implementations:
  - **RequestFirstStatusMatchReport** by means of the SCME;

– **RequestCurrentStatusChangeReport** by means of the SCME;

– **RequestEveryStatusChangeReport** by means of the SCME; and

– monitoring based on issuing by the SCSM of the **RequestReportBCSMEvent** operation and subsequent reception of the **EventReportBCSM** operation to report the availability of the resource. Both the Request and Report occur in a single different call context. In this case, operations to the SDF or equivalent SCF functionality may be used for scanning the status of resources.

• (CE2.23) MidCallEDP: This event is an external event caused by a reception of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-12/Q.1228 – Partial expansion of the state C2.3 FSM for CS

12.5.1.3.2.3.1 State C2.3.1: Preparing CS Instructions

In this state, the FSM for CS prepares the instructions for the SSF to complete the call. The following events are considered in this state:

• (Ce2.3.1) Instruction_Ready: This is an internal event that takes place only when the required resource is available. In this case, the FSM for CSA has obtained the address of the

free resource via the Get_Resource method of the RCO (see 12.4.7). This event maps into the event (Ce2.8).

- (Ce2.3.2) Non-Call_Processing_Instructions: This is an internal event caused by the following cases:

  i) When the service logic needs to send an operation such as the following to the SSF:
     - **ApplyCharging**;
     - **CallInformationRequest**;
     - **FurnishChargingInformation**;
     - **RequestNotificationChargingEvent**;
     - **SendChargingInformation**.

  ii) When the following operations have been sent to the SSF by the associate FSM for CSA instance:
     - **RequestReportBCSMEvent**.

  iii) When the application timer $T_{SCF\text{-}SSF}$ expires. In this case, the FSM for CS sends a **ResetTimer** operation to the SSF for the corresponding CS.

  This event causes a transition back to state C2.3.1, **Preparing CS Instructions**.

- (Ce2.3.3) Busy_Line/Trunk: This is an internal event caused by the RCO when no terminating line/trunk is available. This event causes the HoldCallInNetwork operation to be sent to the SSF, and a transition to state C2.3.2, **Queueing**.

12.5.1.3.2.3.2 State C2.3.2: Queueing

In this state, the FSM for CS is awaiting an indication from the RCO to proceed with routing a call to an idle trunk/line. The support of playing various announcements is also provided when the FSM for CS is in this state. In this Recommendation, the relevant further expansion of the state is not provided; however, it is not different from that of the FSM for CS States C3. Nevertheless, if announcements are completed before the call is dequeued and the SSF FSM has transitioned to state **Waiting for Instructions**, the operation should be sent to set the T_{SSF} with an appropriate value. Once the FSM for CS enters this state, the Queuieng Timer is started. The role of this timer is as follows:

– the Queueing Timer limits the time that a call can spend in the queue, and its value may be customer-specific.

The following events are considered in this state:

- (Ce2.3.4) Idle_Line/Trunk: This is an internal event, which maps into the event (Ce2.8).
- (Ce2.3.5) Queueing_Timer_Expired: This is an internal event, which results in processing the Cancel method of the RCO and maps into the event (Ce2.8) [following procedures depend on the decision of the service logic that may play (or not play) the terminating announcement].

12.5.1.3.2.4 State C2.4: Waiting for Facility

The following events are considered in this state:

- (Ce2.18) First facility Events: When the SLPI requests to send the following operations to the SSF, the FSM will make this transition:
  - **RequestReportFacilityEvent**.
- (Ce2.19) Subsequent Facility Event: When the following operations are received from the SSF, the FSM will make this transition:

  - **RequestReportFacilityEvent**;
  - **SendFacilityInformation**;
  - **EventReportFacility** (not_last_case);
  - **ApplyCharging**;
  - **CallInformationRequest**;
  - **Cancel**(allRequests);
  - **DisconnectLeg**;
  - **HoldCallInNetwork**;
  - **MergeCallSegments**;
  - **MoveLeg**;
  - **RequestNotificationChargingEvent**;
  - **RequestReportBCSMEvent**;
  - **ResetTimer**;
  - **SendChargingInformation**;
  - **SplitLeg**.
- (Ce2.20) last_facility_event: When the following operation is received from the SSF, the FSM will make this transition:
  - **EventReportFacility** (last_facility_event).
- (CE2.22) MidCallEDP: This event is an external event caused by a reception of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".

Other operations are processed only if they do not cause a state transition from C2.4.

This concludes the description of state C2, **Preparing CS instructions**.

12.5.1.3.3 State C3: Suspended and User Interaction

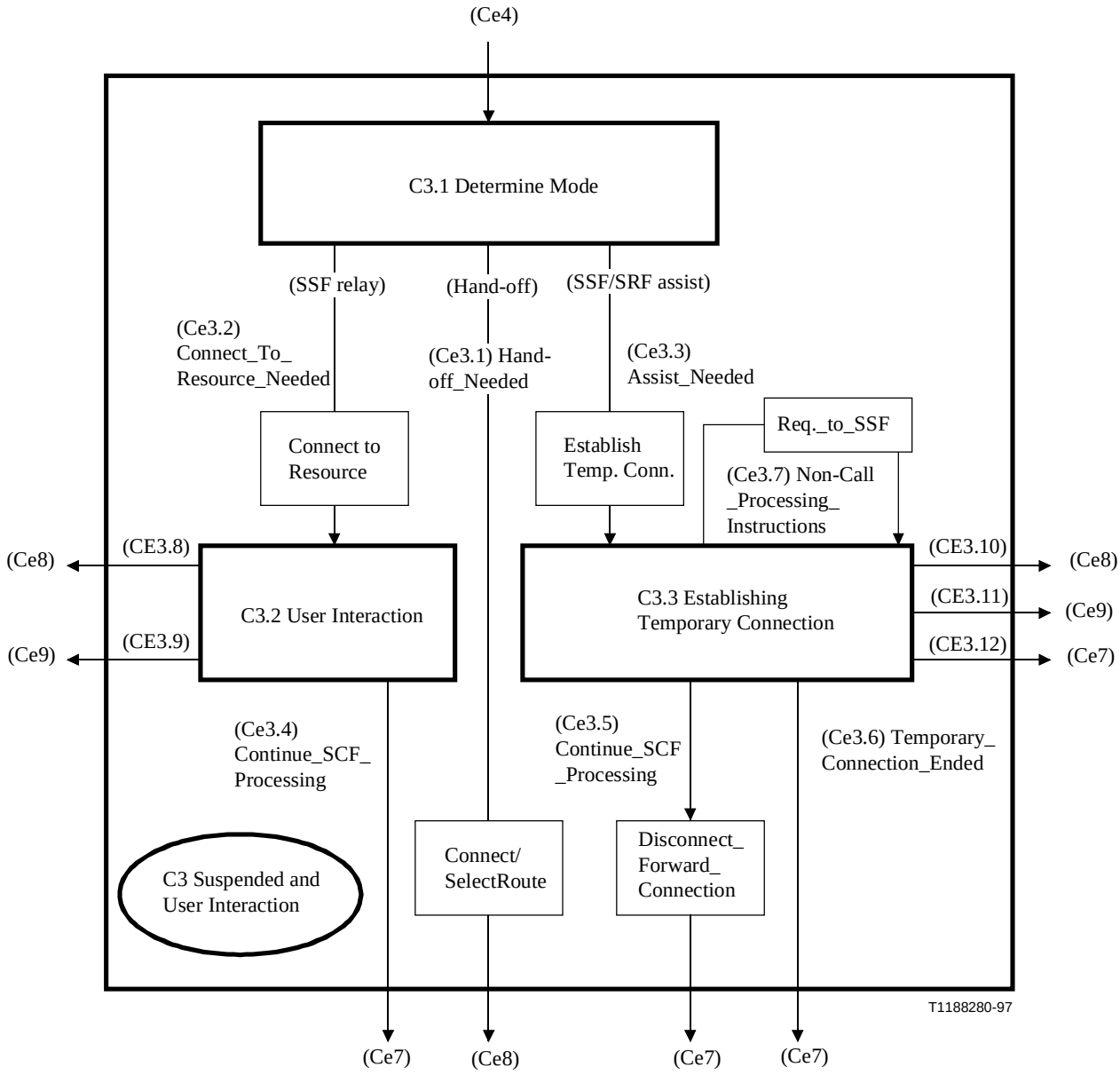
The following events are considered in this state:

- (Ce7) Continue_SCF_Processing: In this case, the SCF has obtained all the information from the SRF, that is needed to instruct the SSF to complete the call. This event causes a transition to state C2, **Preparing CS Instructions**.
- (Ce8) Processing_Completed: This is an internal event, caused by the end of processing for the CS. This event causes a transition to the state C1, **CS Control Idle**.
- (Ce9) CS_Monitoring_Needed: This is an internal event, caused by the necessity of monitoring. This event causes a transition to the state C2.2, **Waiting for Notification or Request**.
- (CE10) MidCallEDP: This event is an external event caused by a reception of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in Figure 12-13.

12.5.1.3.3.1 State C3.1: Determine Mode

The following events are considered in this state:



(CE3.8), (CE3.10): Last_CS_Event_Received
(CE3.9), (CE3.11): Not_Last_CS_Event_Received
(CE3.12): CS_Control_Requested

Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-13/Q.1228 – Partial expansion of the state C3 FSM for CS

- (Ce3.1) Hand-off_Needed: This is an internal event that takes place only with the hand-off case. In this case, the SCF sends the **Connect** or **SelectRoute** operation with the handed-off SSF address to the initiating SSF. This event is mapped as the FSM event (Ce8).
- (Ce3.2) Connect_To_Resource_Needed: This is an internal event that takes place only in the case of initiating SSF relay. In this case, the SCF sends the **ConnectToResource** operation to the initiating SSF. This event causes a transition to the state C3.2, **User Interaction**. In this case, the FSM instance for CS notifies the associate FSM instance for CSA of this event (User_Interaction_Requested).
- (Ce3.3) Assist_Needed: This is an internal event that takes place when either the assisting SSF or the direct SCF-SRF relation is needed. In this case, the SCF sends the

EstablishTemporaryConnection operation to the initiating SSF with the assisting SSF address or the assisting SRF address. This event causes a transition to the state C3.3, **Establishing Temporary Connection**.

12.5.1.3.3.2 State C3.2: User Interaction

When the FSM for CS transits to the other state, it must be notified to the associate FSM for CSA for managing.

The following events are considered in this state:

- (Ce3.4) Continue_SCF_Processing: In this case, the SCF has obtained all the information from the SRF, that is needed to instruct the SSF to complete the call. This event is mapped as the FSM event (Ce7).
- (CE3.8) Last_CS_Event_Received: This is an external event caused by a reception of a last event from the corresponding CS. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).
- (CE3.9) Not_Last_CS_Event_Received: This is an external event. This event causes a transition to the state C2.2, **Waiting for Notification or Request**. This event is mapped as the FSM event (Ce9).

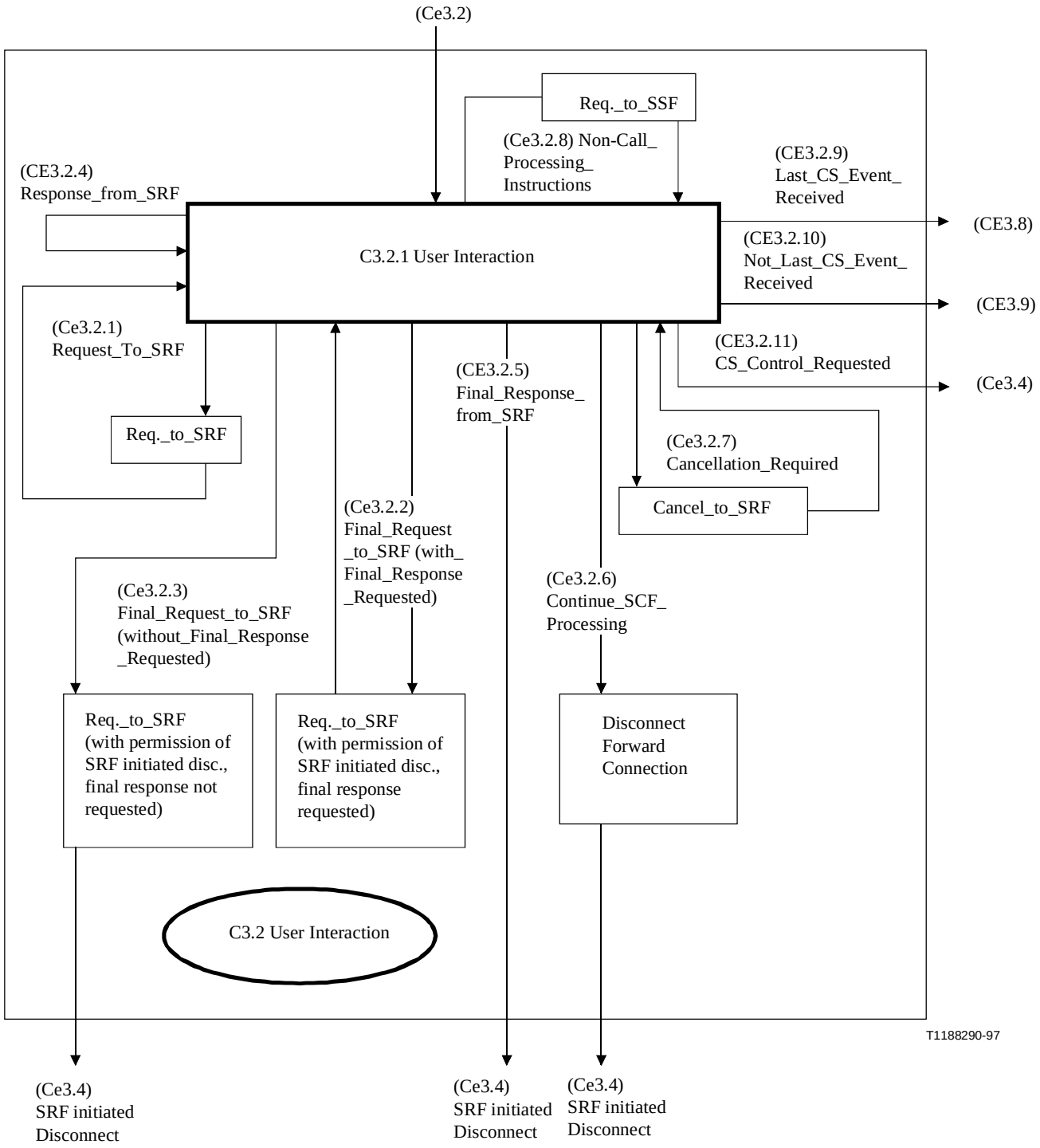
To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in Figure 12-14.

12.5.1.3.3.2.1 State C3.2.1: User Interaction

The following events are considered in this state:

- (Ce3.2.1) Request_To_SRF: This event is an internal event caused by sending one or more of the following operations to the SSF:
  - **PlayAnnouncement**
  - **PromptAndCollectUserInformation**
  - **ScriptRun**
  - **ScriptInfo**
  - **ScriptClose**
  - **PromptAndReceiveMessage**

  This event causes a transition back to the state C3.2.1, **User Interaction**.
- (Ce3.2.2) Final_Request_To_SRF (with Final Response Requested): This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect. In this case, the SCF sends the **PlayAnnouncement** (containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) or **PromptAndCollectUserInformation** operation or ScriptRun, ScriptInformation, ScriptClose, or PromptAndReceiveMessage operation with permission of SRF-initiated disconnect to the SRF. In this case, the FSM for CS transits back to the same state.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-14/Q.1228 – Partial expansion of the state C3.2 FSM for CS

- (Ce3.2.3) Final_Request_To_SRF (without Final Response Requested): This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect, while no **SpecializedResourceReport** operation has been requested to be returned to the SCF when an announcement is completed. In this case, the SCF sends the **PlayAnnouncement** (not containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) or

ScriptRun, **ScriptInformation**, or **ScriptClose** operation with permission of SRF-initiated disconnect to the SRF. This event is mapped as the FSM event (Ce3.4).

- (CE3.2.4) Response_from_SRF: This is an external event caused by the reception of **SpecializedResourceReport,** or **ScriptEvent,** or return result from **PromptAndReceiveMessage,** or return result from **PromptAndCollectUserInformation** operation. On the receipt of either, the FSM for CS transits back to the same state.
- (CE3.2.5) Final_Response_from_SRF: This is an external event caused by the reception of **SpecializedResourceReport,** or **ScriptEvent** or return result from **PromptAndReceiveMessage,** or return result from **PromptAndCollectUserInformation** operation with permission of SRF-initiated disconnect. This event is mapped as the FSM event (Ce3.4).
- (Ce3.2.6) Continue_SCF_Processing: This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and SRF by means of SCF-initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** operation to the initiating SSF. This event is mapped as the FSM event (Ce3.4).
- (Ce3.2.7) Cancellation_Required: This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the SSF. The FSM for CS transits back to the same state.
- (Ce3.2.8) Non-Call_Processing_Instructions: This is an internal event caused by the service logic when there is a need to send one or more of the following operations to the SSF:
  - **ApplyCharging**;
  - **FurnishChargingInformation**;
  - **RequestNotificationChargingEvent**; and
  - **SendChargingInformation**.
- (CE3.2.9) Last_CS_Event_Received: This is an external event caused by a reception of one of the following operations from the SSF:
  - **CallInformationReport**;
  - **ApplyChargingReport**;
  - **EventReportBCSM** (for Abandon/Disconnect EDP-N);
  - **OAbandon** (for EDP-N);
  - **ODisconnect** (for EDP-N);
  - **OMidCall** (for EDP-N);
  - **TAbandon** (for EDP-N);
  - **TDisconnect** (for EDP-N); and
  - **EntityReleased.**

  In this case, there is neither an outstanding armed EDP, a pending **CallInformationReport,** nor a pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (CE3.8).
- (CE3.2.10) Not_Last_CS_Event_Received: This is an external event caused by a reception of one of the following operations from the SSF:
  - **CallInformationReport**;
  - **ApplyChargingReport**;

– **EventReportBCSM** (for Abandon/Disconnect EDP-N);
– **OAbandon** (for EDP-N);
– **ODisconnect** (for EDP-N);
– **OMidCall** (for EDP-N);
– **TAbandon** (for EDP-N); and
– **TDisconnect** (for EDP-N).

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. This event causes a transition to the state C2.2, **Waiting for Notification or Request**. This event is mapped as the FSM event (CE3.9).

- (CE3.2.11) CS_Control_Requested: This is an external event caused by a reception of one of the following operations from the SSF:
  – **EventReportBCSM** (for Abandon/Disconnect EDP-R);
  – **OAbandon** (for EDP-R);
  – **ODisconnect** (for EDP-R);
  – **OMidCall** (for EDP-R);
  – **TAbandon** (for EDP-R); and
  – **TDisconnect** (for EDP-R).

  This event causes a transition to the state C2.1, **Preparing CS Instructions**. This event is mapped as the FSM event (Ce3.4).

It should be noted that the bearer connection between the SSF and the SRF is disconnected when the FSM for CS exits from this state.

12.5.1.3.3.3 State C3.3: Establishing Temporary Connection

The following events are considered in this state:

- (Ce3.5) Continue_SCF_Processing: This is an internal event that takes place when the FSM instance for CS finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the assisting SSF/SRF by means of SCF-initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** operation to the initiating SSF. This event is mapped as the FSM event (Ce7). In this case, the FSM instance for CS notifies the associate FSM instance for CSA of this event (User_Interaction_Finished).
- (Ce3.6) Temporary_Connection_Ended: This is an internal event caused by the notification from the associate FSM instance for CSA because of the end of the user interaction for the assisting SRF. This event is mapped as the FSM event (Ce7).
- (Ce3.7) Non-Call_Processing_Instructions: This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one or more of the following operations to be issued to the SSF:
  – **ApplyCharging**;
  – **FurnishChargingInformation**;
  – **RequestNotificationChargingEvent**; and
  – **SendChargingInformation**.

  The FSM for CS transits back to the same state.

- (CE3.10) Last_CS_Event_Received: This is an external event caused by a reception of one of the following operations from the SSF:
  - **CallInformationReport**;
  - **ApplyChargingReport**;
  - **EventReportBCSM** (for Abandon/Disconnect EDP-N);
  - **OAbandon** (for EDP-N);
  - **ODisconnect** (for EDP-N);
  - **OMidCall** (for EDP-N);
  - **TAbandon** (for EDP-N);
  - **TDisconnect** (for EDP-N); and
  - **EntityReleased**.

  In this case, there is neither an outstanding armed EDP, a pending **CallInformationReport,** nor a pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).
- (CE3.11) Not_Last_CS_Event_Received: This is an external event caused by a reception of one of the following operations from the SSF:
  - **CallInformationReport**;
  - **ApplyChargingReport**;
  - **EventReportBCSM** (for Abandon/Disconnect EDP-N);
  - **OAbandon** (for EDP-N);
  - **ODisconnect** (for EDP-N);
  - **OMidCall** (for EDP-N);
  - **TAbandon** (for EDP-N); and
  - **TDisconnect** (for EDP-N).

  In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. This event causes a transition to the state C2.2, **Waiting for Notification or Request**. This event is mapped as the FSM event (Ce9).
- (CE3.12) CS_Control_Requested: This is an external event caused by a reception of one of the following operations from the SSF:
  - **EventReportBCSM** (for Abandon/Disconnect EDP-R);
  - **OAbandon** (for EDP-R);
  - **ODisconnect** (for EDP-R);
  - **OMidCall** (for EDP-R);
  - **TAbandon** (for EDP-R); and
  - **TDisconnect** (for EDP-R).

  This event causes a transition to the state C2.1, **Preparing CS Instructions**. This event is mapped as the FSM event (Ce7).

12.5.1.4 Finite State Model for Specialized Resource

Figure 12-15 shows the general State Diagram of the FSM for Specialized Resource as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

12.5.1.4.1 State R1: SRF Control Idle

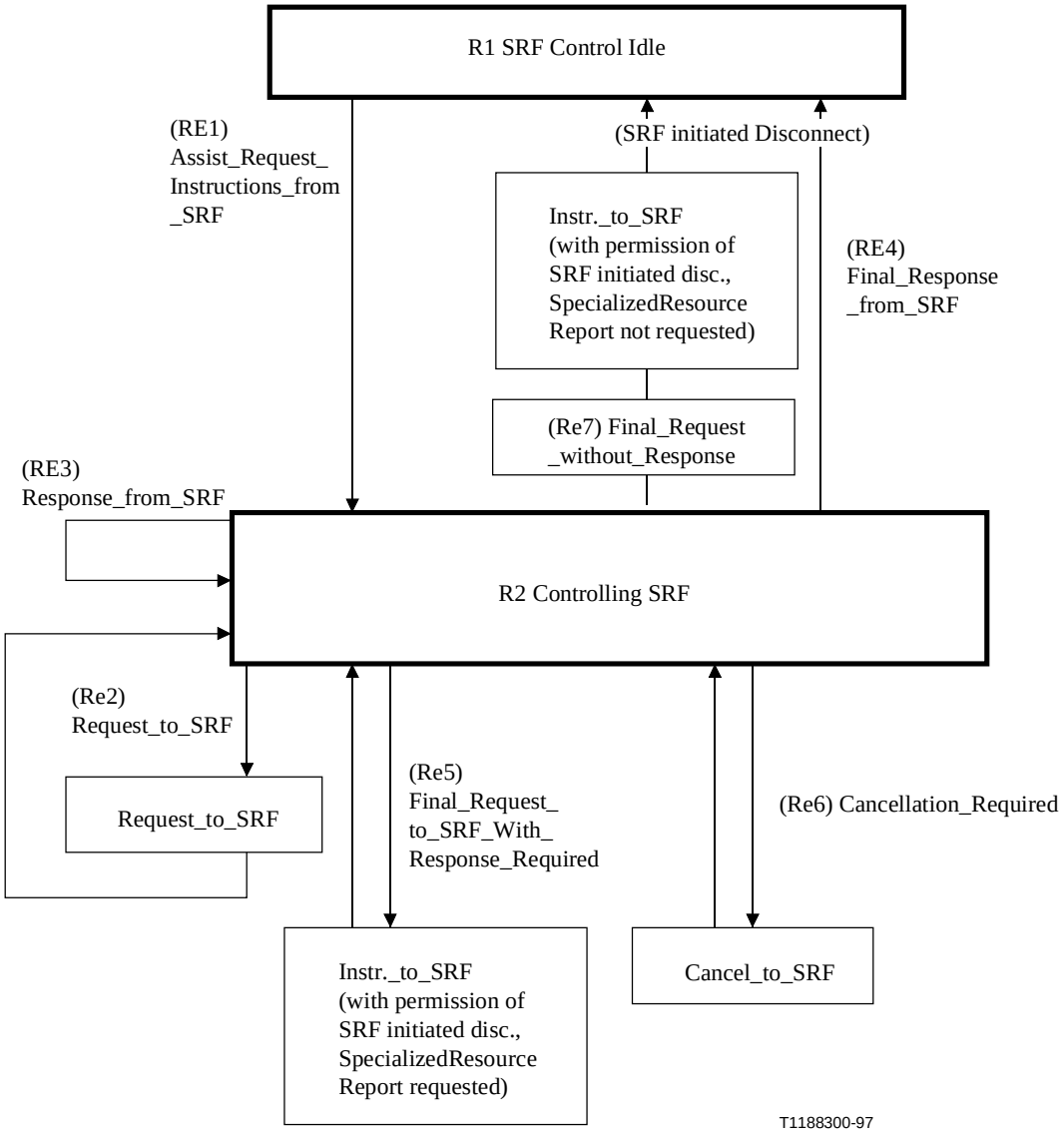
The following event is considered in this state:

- (RE1) Assist_Request_Instructions_from_SRF: This is an external event caused by a reception of an **AssistRequestInstructions** operations from the SRF. This event causes a transition to the state R2, **Controlling SRF**.

12.5.1.4.2 State R2: Controlling SRF

The following events are considered in this state:

- (Re2) Request_to_SRF: This is an internal event caused by the SLPI when there is a need to send one or more of the following operations to be issued to the SSF:



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-15/Q.1228 – FSM for specialized resource

– **PlayAnnouncement**;
– **PromptAndCollectUserInformation**;
– **ScriptRun**;
– **ScriptInfo**;
– **ScriptClose**; and
– **PromptAndReceiveMessge**.

This event causes a transition back to the same state.

- (RE3) Response_from_SRF: This is an external event caused by a reception of one of the following:
  – **SpecializedResourceReport**;
  – **Return Result for PromptAndCollectUserInformation**;
  – **PromptAndReceiveMessage Result**; and
  – **ScriptEvent**.

  This event causes a transition back to the same state.
- (RE4) Final_Response_from_SRF: This is an external event caused by a reception of one of the followings after SRF-initiated disconnect:
  – **SpecializedResourceReport**;
  – **Return Result for PromptAndCollectUserInformation**;
  – **PromptAndReceiveMessage Result**; and
  – **ScriptEvent**.

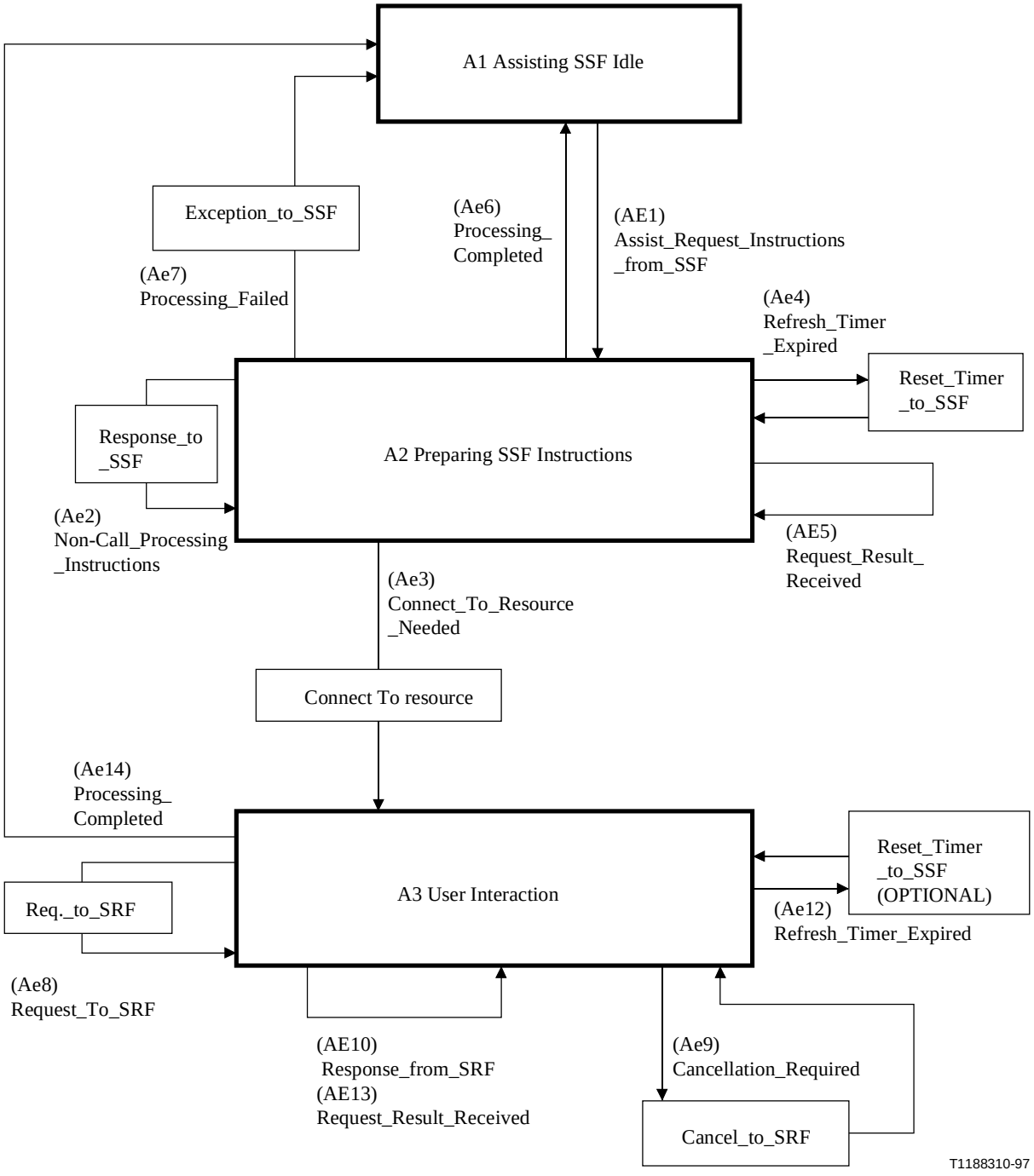
  This event causes a transition to the state R1, **SRF Control Idle**.
- (Re5) Final_Request_to_SRF_with_Response_Required: This is an internal event caused by the SLPI when there is a need to send one of the following operations with permission of SRF-initiated disconnect to be issued to the SSF:
  – **ScriptRun**;
  – **PromptAndReceiveMessage**;
  – **PlayAnnouncement**; and
  – **PromptAndCollectUserInformation**.

  This event causes a transition back to the same state.
- (Re6) Cancellation_Required: This is an internal event that takes place when the SLPI cancels the previous **Play Announcement** or **PromptAndCollectUserInformation** operation. In this case, the SCSM sends the Cancel operation to the SRF, and transits back to the same state.
- (Re7) Final_Request_without_Response: This is an internal event that takes place when the SCSM finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-Initiated disconnect, while no **SpecializedResourceReport** operation is requested to be returned to the SCF in case an announcement is completed. In this case, the SCF sends the **PlayAnnouncement** (not containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) with permission of SRF-initiated disconnect to the SRF. **ScriptRun**, **ScriptClose** and **ScriptInformation** operations are also valid at this event. This event causes a transition to the state R1, **SRF Control Idle**.

12.5.1.5 Finite State Model for Assisting SSF

Figure 12-16 shows the general State Diagram of the FSM for assisting SSF as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

The FSM for assisting SSF has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, T_{SSF}, to guard the association between the assisting SSF and the SCF.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-16/Q.1228 – FSM for Assisting SSF

Timer $T_{SCF-SSF}$ is set in the following cases:

– when the SCF receives an **AssistRequestInstructions** operation. In this case, this timer is restart when a first request, other than **ResetTimer** operation, is sent to the assisting SSF. On the expiration of timer $T_{SCF-SSF}$, the FSM for assisting SSF may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF-SSF}$. On the second expiration of $T_{SCF-SSF}$, the FSM for assisting SSF informs the SLPI and the maintenance functions, and the FSM for assisting SSF transitions to the state A1, **Assisting SSF Idle**;

– when FSM for assisting SSF enters the "User Interaction" state. In this case, on the expiration of $T_{SCF-SSF}$, the SCF may restart T_{SSF} using the **ResetTimer** operation any number of times (OPTIONAL).

NOTE – The word "OPTIONAL" refers to the use of the application timer $T_{SCF-SSF}$. Whether it is used depends on an implementation, but, if used, it must be synchronized with T_{SSF} in the assisting SSF FSM.

In all two cases, $T_{SCF-SSF}$ may respectively have two different values as defined by the application. The values of $T_{SCF-SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF-SSF}$.

12.5.1.5.1 State A1: Assisting SSF Idle

The following event is considered in this state:

• (AE1) Assist_Request_Instructions_from_SSF: This is an external event caused by a reception of the following operation:

  – **AssistRequestInstructions**

  This event causes a transition to the state A2, **Preparing SSF Instructions**.

12.5.1.5.2 State A2: Preparing SSF Instructions

The following events are considered in this state:

• (Ae2) Non-Call_Processing_Instructions: This is an internal event caused by the SLPI when there is a need to send such an operation to the assisting SSF. It causes one or more of the following operations to be issued to the SSF:

  – **ApplyCharging**;

  – **FurnishChargingInformation**; and

  – **SendChargingInformation**.

  This event causes a transition back to the state A2, **Preparing SSF Instructions**.

• (Ae3) Connect_To_Resource_Needed: This is an internal event. In this case, the SCF sends the **ConnectToResource** operation to the assisting SSF. This event causes a transition to the state A3, **User Interaction**.

• (Ae4) Refresh_Timer_Expired: This is an internal event, which results in sending the **ResetTimer** operation to the assisting SSF and a transition back to the same state.

• (AE5) Request_Result_Received: This is an external event caused by a reception of the following operation from the assisting SSF:

  – **ApplyChargingReport**

  This event causes a transition to the same state.

• (Ae6) Processing_Completed: This is an internal event, caused by the end of processing for the assisting SSF. This event causes a transition to the state A1, **Assisting SSF Idle**.

- (Ae7) Processing_Failed: This (internal) event causes an appropriate exception processing and a transition back to the state A1, **Assisting SSF Idle**.

12.5.1.5.3 State A3: User Interaction

The following events are considered in this state:

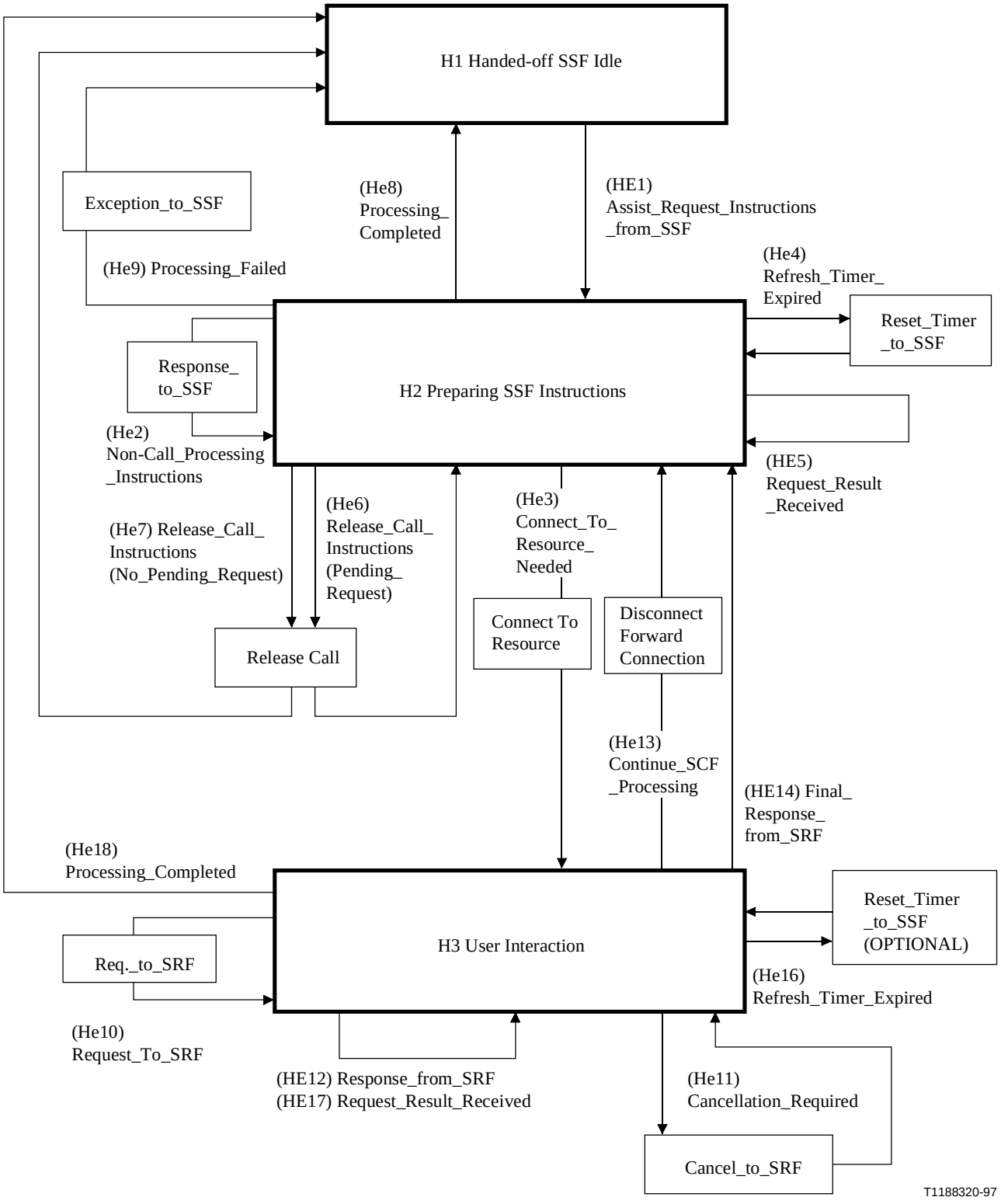
- (Ae8) Request_To_SRF: This event is an internal event caused by sending one or more of the following operations to the assisting SSF.
  - **PlayAnnouncement**;
  - **PromptAndCollectUserInformation**;
  - **ScriptRun**;
  - **ScriptInfo**;
  - **ScriptClose**; and
  - **PromptAndReceiveMessage**.

  This event causes a transition back to the state A3, **User Interaction**.
- (Ae9) Cancellation_Required: This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the assisting SSF. The FSM for assisting SSF transits back to the same state.
- (AE10) Response_from_SRF: This is an external event caused by the reception of **SpecializedResourceReport** or return result from **PromptAndCollectUserInformation** operation or PromptAndReceiveMessage Result or ScriptEvent. On the receipt of either, the FSM for assisting SSF transits back to the same state.
- (Ae12) Refresh_Timer_Expired: This is an internal event, which results in sending the **ResetTimer** operation to the assisting SSF and a transition back to the same state.
- (AE13) Request_Result_Received: This is an external event caused by a reception of the following operation from the assisting SSF:
  - **ApplyChargingReport**

  This event causes a transition to the same state.
- (Ae14) Processing_Completed: This is an internal event, caused by the end of processing for the assisting SSF. This event causes a transition to the state A1, **Assisting SSF Idle**.

12.5.1.6 Finite State Model for Handed-off SSF

Figure 12-17 shows the general State Diagram of the FSM for handed-off SSF as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses. The Hand-Off FSM for IN CS-2 applies only to the case where final treatment is to be applied.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-17/Q.1228 – FSM for Handed-off SSF

The FSM for handed-off SSF has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, T_{SSF}, to guard the association between the handed-off SSF and the SCF.

Timer $T_{SCF-SSF}$ is set in the following cases:

– when the SCF receives an **AssistRequestInstructions** operation. In this case, this timer is restart when a first request, other than **ResetTimer** operation, is sent to the handed-off SSF. On the expiration of timer $T_{SCF-SSF}$, the FSM for handed-off SSF may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF-SSF}$. On the second expiration of $T_{SCF-SSF}$, the FSM for handed-off SSF informs the SLPI and the maintenance functions, and the FSM for handed-off SSF transitions to the state H1, **Handed-off SSF Idle**;

– when FSM for handed-off SSF enters the "**User Interaction**" state. In this case, on the expiration of $T_{SCF-SSF}$, the SCF may restart T_{SSF} using the **ResetTimer** operation any number of times (OPTIONAL).

NOTE – The word "OPTIONAL" refers to the use of the application timer $T_{SCF-SSF}$. Whether it is used depends on an implementation, but, if used, it must be synchronized with T_{SSF} in the handed-off SSF FSM.

In all two cases, $T_{SCF-SSF}$ may respectively have two different values as defined by the application. The values of $T_{SCF-SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF-SSF}$.

12.5.1.6.1 State H1: Handed-off SSF Idle

The following event is considered in this state:

• (HE1) Assist_Request_Instructions_from_SSF: This is an external event caused by a reception of the following operation:

  – **AssistRequestInstructions**

  This event causes a transition to the state H2, **Preparing SSF Instructions**.

12.5.1.6.2 State H2: Preparing SSF Instructions

The following events are considered in this state:

• (He2) Non-Call_Processing_Instructions: This is an internal event caused by the SLPI when there is a need to send such an operation to the handed-off SSF. It causes one or more of the following operations to be issued to the SSF:

  – **ApplyCharging**;

  – **FurnishChargingInformation**; and

  – **SendChargingInformation**.

  This event causes a transition back to the state H2, **Preparing SSF Instructions**.

• (He3) Connect_To_Resource_Needed: This is an internal event. In this case, the SCF sends the **ConnectToResource** operation to the handed-off SSF. This event causes a transition to the state H3, **User Interaction**.

• (He4) Refresh_Timer_Expired: This is an internal event, which results in sending the **ResetTimer** operation to the handed-off SSF and a transition back to the same state.

• (HE5) Request_Result_Received: This is an external event caused by a reception of the following operation from the handed-off SSF:

  – **ApplyChargingReport**

  This event causes a transition to the same state.

• (He6) Release_Call_Instructions (Pending_Request): This is an internal event caused by sending a ReleaseCall operation when there are one or more pending requests

(CallInformationRequest or ApplyCharging). This event causes a transition to the same state.

- (He7) Release_Call_Instructions (No_Pending_Request): This is an internal event caused by sending a ReleaseCall operation when there are no pending requests (neither CallInformationRequest nor ApplyCharging). This event causes a transition to the state H1, Handed-off SSF Idle.
- (He8) Processing_Completed: This is an internal event, caused by the end of processing for the handed-off SSF. This event causes a transition to the state H1, **Handed-off SSF Idle**.
- (He9) Processing_Failed: This (internal) event causes an appropriate exception processing and a transition back to the state H1, **Handed-off SSF Idle**.

12.5.1.6.3 State H3: User Interaction

The following events are considered in this state:

- (He10) Request_To_SRF: This event is an internal event caused by sending one or more of the following operations to the handed-off SSF.
  - **PlayAnnouncement**;
  - **PromptAndCollectUserInformation**;
  - **ScriptRun**;
  - **ScriptInfo**;
  - **ScriptClose**, and
  - **PromptAndReceiveMessage**.

  This event causes a transition back to the state H3, **User Interaction**.
- (He11) Cancellation_Required: This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the handed-off SSF. The FSM for handed-off SSF transits back to the same state.
- (HE12) Response_from_SRF: This is an external event caused by the reception of **SpecializedResourceReport** or return result from **PromptAndCollectUserInformation** operation or PromptAndReceiveMessage Result or ScriptEvent. On the receipt of either, the FSM for handed-off SSF transits back to the same state.
- (He13) Continue_SCF_Processing: This is an internal event that takes place when the FSM instance for handed-off SSF finishes the user interaction and requests the disconnection of bearer connection between the handed-off SSF and SRF by means of SCF-initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation to the handed-off SSF. The FSM for handed-off SSF transits to the state H2, **Preparing SSF Instructions**.
- (HE14) Final_Response_from_SRF: This is an external event caused by the reception of **SpecializedResourceReport** or return result from **PromptAndCollectUserInformation** operation or PromptAndReceiveMessage Result or ScriptEvent with the permission of SRF-initiated disconnect. The FSM for handed-off SSF transits to the state H2, **Preparing SSF Instructions**.
- (He16) Refresh_Timer_Expired: This is an internal event, which results in sending the **ResetTimer** operation to the handed-off SSF and a transition back to the same state.

- (HE17) Request_Result_Received: This is an external event caused by a reception of the following operation from the handed-off SSF:

  – **ApplyChargingReport**

  This event causes a transition to the same state.

- (He18) Processing_Completed: This is an internal event, caused by the end of processing for the handed-off SSF. This event causes a transition to the state H1, **Handed-off SSF Idle**.

12.5.2 SDF-related states (SCSM-SDF)

The interaction with the SDF is possible from any state of the SCF. In the following subclauses, the SDF-related states are specified. The model describes the relationship of one SCF with one SDF. If an SCF needs to access another SDF, a new FSM should be instantiated.

In what follows, the states and events are enumerated independent of the rest of the SCSM; the discussion is accompanied by Figure 12-18.

12.5.2.1 State 1: Idle

The following event is considered in this state:

- (e1) Bind_Request: This is an internal event, caused by the need of the service logic to create an association with an SDF in order to begin accessing data. This event causes a transition to the state 2, **Wait for subsequent requests**.

12.5.2.2 State 2: Wait for subsequent requests

In this state, subsequent operations to be sent with the **Bind** operation (in the same message) to the SDF are expected. The following two events are considered in this state:

- (e2) Request_to_SDF: This is an internal event caused by the reception of an operation. The operation is buffered until the reception of a delimiter (or a timer expiration). The SCSM remains in the same state; and
- (e3) Request_to_SDF_with_Bind: This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing the argument of the **Bind** operation and other operations' arguments, if any, is sent to the SDF. This event causes a transition out of this state to the State 3, **Wait for Bind result**.

12.5.2.3 State 3: Wait for Bind result

In this state, the SCF is waiting for the response from the SDF. Two events are considered in this state:

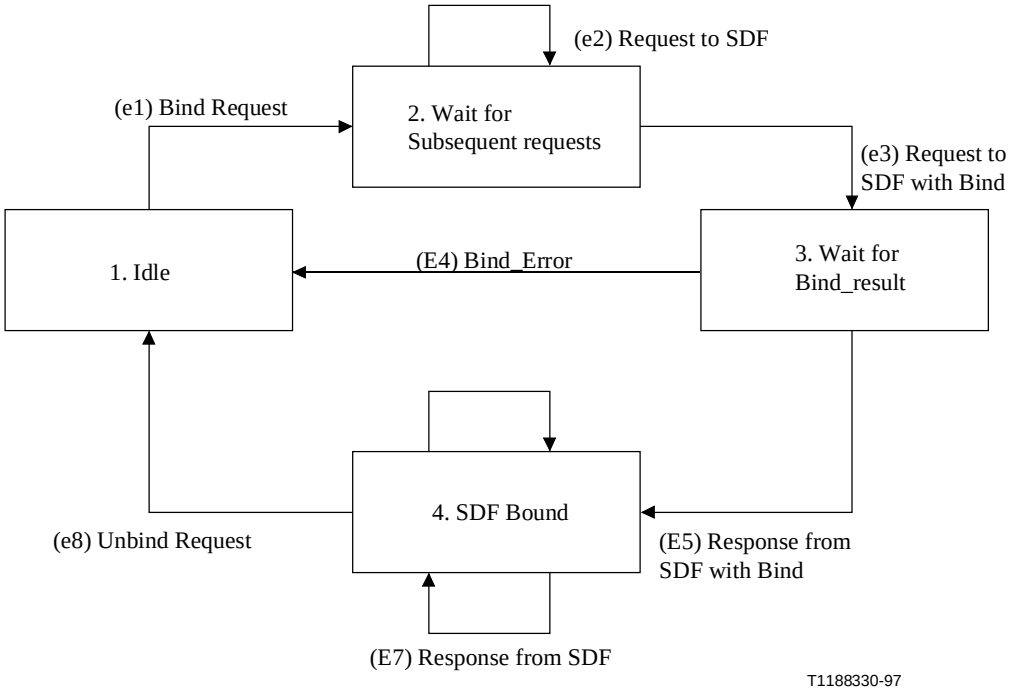
- (E4) Bind_Error: This is an external event, caused by the reception of an error to the **Bind** operation previously issued to the SDF. This event causes a transition out of this state to state 1, **Idle**.
- (E5) Response_from_SDF_with_Bind: This is an external event, caused by the reception of a **Bind** result combined with the responses to other operations previously issued to the SDF (if any). This event causes a transition to state 4, **SDF Bound**.

12.5.2.4 State 4: SDF Bound

In this state, the SCF has established an authenticated access to the SDF, and is waiting for requests to the SDF from the service logic or is waiting for responses to the operations previously issued to the SDF. Three events are considered in this state:

- (e6) Request_to_SDF: This is an internal event, caused by the need of the service logic to access data in the SDF. The SCSM remains in the same state;
- (E7) Response_from_SDF: This is an external event, caused by the reception of responses to the operations previously issued to the SDF. The SCSM remains in the same state; and
- (e8) Unbind_request: This is an internal event, caused by the need of the service logic to terminate the authenticated access to the SDF. This event causes a transition state to state 1, **Idle**.

In addition to the above model, an SDL description of SCSM is shown in Annex A.8.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-18/Q.1228 – The SDF-related states

12.5.3 SCF-related states

The interaction with another SCF is possible from any state of the invoking SCF. In the following subclauses, the SCF-related states are specified. The models describe the relationship of one SCF with another SCF on both sides of the relationship. If an SCF needs to access another SCF, a new FSM should be instantiated.

In what follows, the states and events are numbered independently of the rest of the SCSM. A first subclause describes the SCF FSM when the SCF has initiated a dialogue with another SCF. A second subclause describes the other part of the SCF-SCF relationship, i.e. when the SCF is responding to a request from another SCF.

12.5.3.1 Controlling SCF FSM (SCSM-Con)

12.5.3.1.1 State 1: Idle

The following event is considered in this state:

- (e1) SCF Bind_request : This is an internal event, caused by the need of the service logic to establish a relationship with another SCF and to receive cooperation from it, to further proceed with a call. This event causes a transition to state 2, **Preparing Handling Information Request**. It causes – SCFBind operation – to be issued to the supporting SCF.

12.5.3.1.2 State 2: Preparing Handling Information Request

In this state, the SCF is preparing a request to the supporting SCF to be assisted in the processing of the call. The following events are considered in this state:

- (e2) Send Handling Information Request sent: It causes a Handling Information request operation to be sent the supporting SCF. This event causes a transition to state 3: **Waiting for Bind Result**;
- (E24) SCF Bind error: This is an external event caused by SCFBind operation error reception. It causes a transition back to state 1, **Idle**.
- (e25) SCFUnbind: This is an internal event causing sending of SCFUnbind (e.g. because the SCFBind result timer is expired). It causes a transition back to state 1, **Idle**.

12.5.3.1.3 State 3: Waiting for Bind Result

In this state, the SCF is waiting the result to Bind request. The following events are considered in this state:

- (E3) SCF Bind error: This is an external event caused by the reception of the error result of the previously issued Bind operation. It causes a transition back to state 1, **Idle**.
- (E4) SCF Bind successful: This is an external event caused by the reception of the successful result of the previously issued Bind operation. It causes a transition to state 4, **Assisted Mode**.
- (e26) Timer expiration: This is an internal event caused by the expiration of a guard timer. It causes a transition back to state **Preparing SCF Unbind Request**.

12.5.3.1.4 State 4: Assisted Mode

In this state, the SCF has sent a request to the supporting SCF to be assisted in the processing of the call. A relationship has been created between the two SCFs. They are cooperating to provide functionalities needed to process a call. The following events can occur in this state:

- (E5) Assist_Completed: This is an external event caused by the reception of the result of a previously issued operation. The result indicates that the SCF can continue with the processing of the call and that the assistance of an assisting SCF is not further required. This event causes a transition to the state 7, **Preparing SCF Unbind request**; unless there is a result pending, in which case it stays in state 4, **pending the receipt of the requested result**.

  This event is caused by a reception of the following operation:

  – **HandlingInformationResult**
- (e6) End_Assist: This is an internal event caused by the need of the service logic to stop its relationship with the supporting service logic. This event causes a transition to the state 7, **Preparing SCF Unbind request**.

- (E7) Additional_Information_Required_from_Supporting_SCF: This is an external event caused by the reception of an operation from the supporting SCF requesting additional information to be able to assist the controlling SCF. This event causes a transition out of this state to state 5 , **Preparing Additional Information**. This event is caused by a reception of a following operation:
  - **ProvideUserInformation**
  - **NetworkCapabilityRequest**
- (e8) Notification_Provided_to_Supporting_SCF_without_Confirmation_Request: This is an internal event caused by the need of the service logic to provide notification for the supporting SCF without the confirmation request. This event causes a transition to the same state, state 4 , **Assisted Mode**. It causes one of the following operations to be issued to the supporting SCF:
  - **Notification-Provided**
  - **ReportChargingInformation**
- (e21) Notification_Provided_to_Supporting_SCF_with_Confirmation_Request: This is an internal event caused by the need of the service logic to provide notification for the supporting SCF with the confirmation request. The controlling SCF waits for the confirmation from the supporting SCF, and this event causes a transition out of this state to state 6, **Waiting for Response from Supporting SCF**. It causes one of the following operations to be issued to the supporting SCF:
  - **ConfirmedReportChargingInformation**
  - **ConfirmedNotification Provided**
- (e22) Handling Information Request sent: This is an internal event caused by the need of the service logic, having received the previously requested **Handling-Information-Result**, to request additional information to the supporting SCF. It causes the following operation to be issued to the supporting SCF:
  - **HandlingInformationRequest**

  The controlling SCF waits for the response from the supporting SCF, and this event does not cause any transition out of this state.
- (E23) Confirmation Provided: This is an external event caused by the reception of one of the following operation results:
  - **Result of ConfirmedReportChargingInformation**
  - **Result of ConfirmedNotification Provided**

  It does not cause any transition out of this state.
- (E19) Establish Charging Record: This is an external event caused by the reception of the EstablishChargingRecord operation. It does not cause any transition out of this state.
- (E20) Request Notification: This is an external event caused by the reception of the RequestNotification operation. It does not cause any transition out of this state.
- (E24) Referral from Supporting SCF: This is an external event caused by the reception of a Referral error as the response to the **Handling-Information-Request** operation. This event is communicated to the internal service logic and does not cause a state transition out of this state.

12.5.3.1.5 State 5: Preparing Additional Information

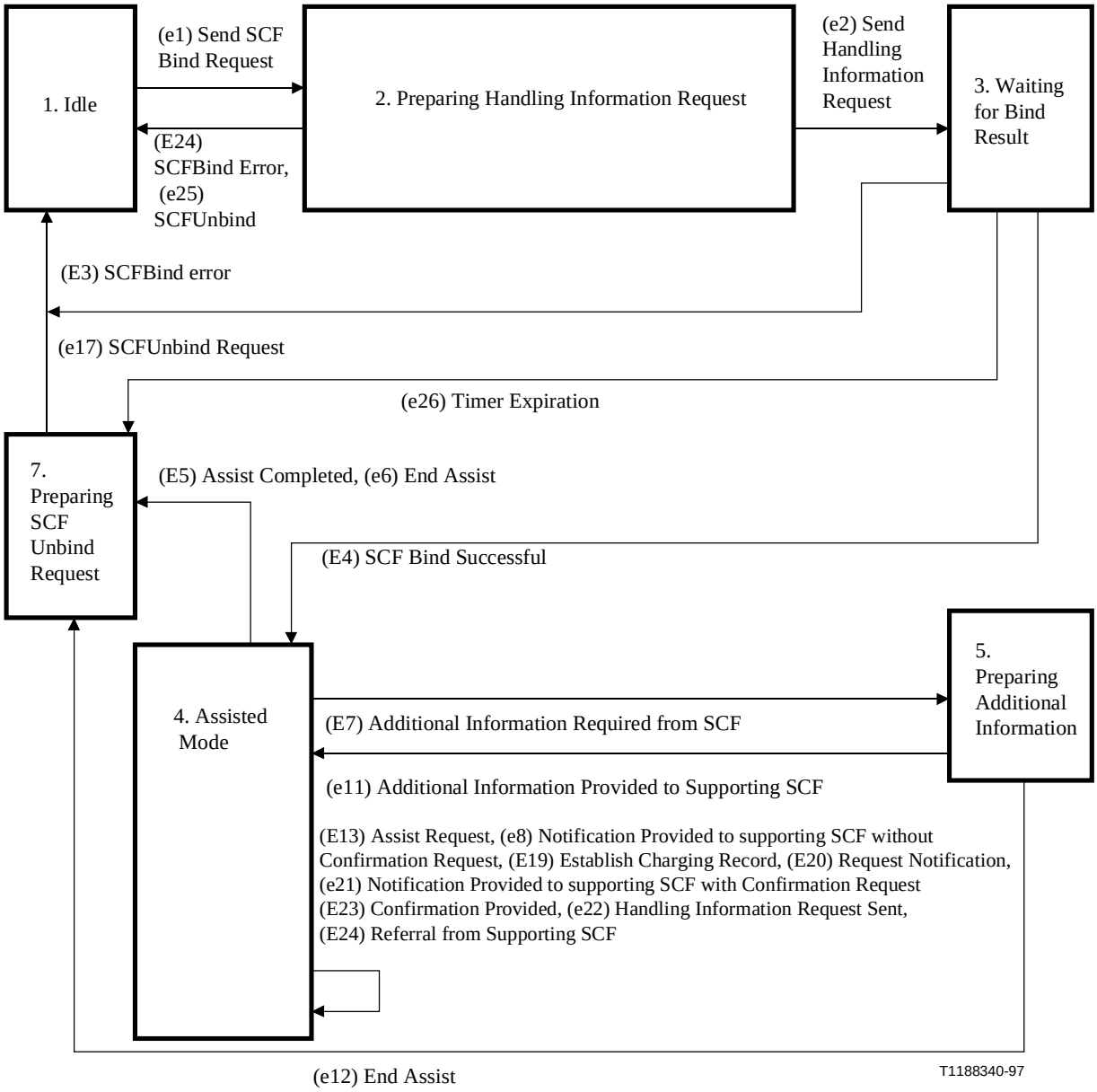
In this state, the SCF is preparing additional information for the supporting SCF so that it can be assisted in the processing of the call. Two events are considered in this state:

- (e11) Additional_Information_Provided_to_Supporting_SCF: This is an internal event. The SCF has collected all the information needed from the supporting SCF and sends the result to the previously issued operation requesting additional information. This event causes a transition to state 4, **Assisted Mode**. It causes the return result of the following operation to be issued to the supporting SCF:
  - **Result of ProvideUserInformation**
  - **Result of NetworkCapability**
- (e12) End_Assist: This is an internal event caused by the need of the service logic to stop its cooperation with the supporting service logic. This event causes a transition back to the state 7, **Preparing SCF Unbind request**.

12.5.3.1.6 State 7: Preparing SCF Unbind request

In this state, the SCF is preparing to send a SCF Unbind operation, to terminate the relationship with the supporting SCF. It causes the following event:

- (e17) SCFUnbind request: This is an internal event that occurs when SCFUnbind operation is sent. This closes the relationship and causes a transition back to the state 1, **Idle**.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-19/Q.1228 – SCSM-Con (Controlling SCF FSM)

12.5.3.2 Supporting SCF FSM (SCSM-Sup)

12.5.3.2.1 State 1: Idle

In this state the SCF is waiting for request from other SCFs. No SLPI is started yet. Only the following event is accepted in this state:

- (E1) SCF Bind Request Received: This is an external event caused by the reception of a SCFBind request operation from the controlling SCF for providing assistance in the processing of the call. This event causes a transition out of this state to state 2, **Processing SCFBind**.

12.5.3.2.2 State 2: Processing SCF Bind

In this state, the SCF is processing SCFBind operation. The following events are considered in this state:

- (e2) Send positive SCFBind result: This is an internal event causing the sending of a positive SCFBind result. It does not cause a state transition.
- (e3) SCF Bind Refused: This is an internal event that occurs when the supporting SCF does not establish a relationship with the controlling SCF. It causes an error SCFBind result to be sent to the controlling SCF. This event causes a transition out of this state to state 1, **Idle**.
- (e24) Timer expiration: This is an internal event caused by guard timer expiration. It does cause a transition back to state 1, **Idle**.
- (E4) Request from controlling SCF: This is an external event caused by the reception of **HandlingInformationRequest** operation. The **HandlingInformationRequest** operation is queued since the supporting SCF service logic is involved in the SCFBind processing (it will be processed as soon as the supporting SCF FSM moves to **Assisting Mode** state). It does not cause a transition out of the state.
- (e28) SCFBindAccepted & Handling Information Request Received: This is an internal event caused by the fact that the **SCFBind** operation is accepted and that a **HandlingInformationRequest** operation has been received. It does not cause any operation to be sent. It causes a transition to **Assisting Mode** state.
- (E11) SCF Unbind Request received: This is an external event caused by the reception of a request from controlling SCF to end the relationship through **SCFUnbind** operation. This event causes a transition back to state 1, **Idle**.

12.5.3.2.3 State 3: Assisting mode

In this state, the SCF is sending operations to provide assistance to the controlling SCF and receives indications from the controlling SCF. The following events are considered in this state.

- (e5) Assist_Provided: This is an internal event caused by the sending of a response to the assistance request previously received from the controlling SCF, provided that the SCF has not foreseen any further assistance. There is no state transition. It causes the following operation to be issued to the controlling SCF:
  - **HandlingInformationResult**
- (E6) SCF Unbind request received: This is an external event caused by the reception of an SCF Unbind request from controlling SCF. It gives an indication from the controlling SCF that the SCF-SCF relationship needs to be ended. This event causes a transition back to state 1, **Idle**.
- (e7) Information_Needed_from_Controlling_SCF: This is an internal event caused by the need of the service logic to receive additional information from the controlling SCF to provide assistance. This event causes a transition out of this state to state 4, **Waiting for Additional Information**. It causes the following operation to be issued to the controlling SCF:
  - **ProvideUserInformation**
  - **NetworkCapabilityRequest**
- (E8) Notification_Provided_by_Controlling_SCF_without_Confirmation_Request: This is an external event caused by the receipt of the notification which was requested to the controlling SCF. This event causes a transition to the same state, state 3, **Assisting Mode**. This event is caused by a reception of the following operations:

– **ReportChargingInformation**

– **Notification-Provided**

- (E20) Notification_Provided_by_Controlling_SCF_with_Confirmation: This is an external event caused by the reception of one of the following operations:

  – **ConfirmedReportChargingInformation**

  – **ConfirmedNotification-Provided**

  This event does not cause a transition out of this state.

- (e12) Notification Requested: This is an internal event caused by the sending of a RequestNotification-operation to the controlling SCF. This event causes a transition to the same state, state 3, **Assisting Mode**.
- (e13) Charging Information: This is an internal event caused by the sending of a EstablishChargingRecord operation to the controlling SCF. This event causes a transition to the same state, state 3, **Assisting Mode**.
- (e21) ConfirmationProvided: This is an internal event caused by the need of the service logic to respond to the confirmation previously received request. It causes one of the following operations to be issued to the controlling SCF:

  – **ReportChargingInformationConfirmation**

  – **Notification-Provided-Confirmation**

  This event does not cause a transition out of **Assisting mode** state.

- (e17) Additional_Information_Result: This is an internal event caused by the sending of a response to the previously received request from the controlling SCF. It causes the following operation to be issued to the controlling SCF:

  – **HandlingInformationResult**

  This event does not cause a transition out of this state 3, **Assisting Mode**.

- (e22) Timer expiration: This is an internal event that occurs when no operation has been received from the controlling SCF for an appropriate period of time. The supporting SCF determines that the relation is no longer in existence and moves back to **Idle** state.
- (E25) Received Handling Information Request: This is an external event caused by the reception of a new **HandlingInformationRequest** operation. It can be accepted only if there is no previous **HandlingInformationResult** operation pending. This event does not cause a state transition out of this state.
- (e31) Referral to controlling SCF: This is an internal event caused by the decision to instruct the controlling SCF to send the pending **HandlingInformationRequest** to another SCF for processing. A Referral error for the **Handling-Information-Request** operation is sent to controlling SCF. It does not cause a transition out of **Assisting Mode** state.

12.5.3.2.4 State 4: Waiting for Additional Information

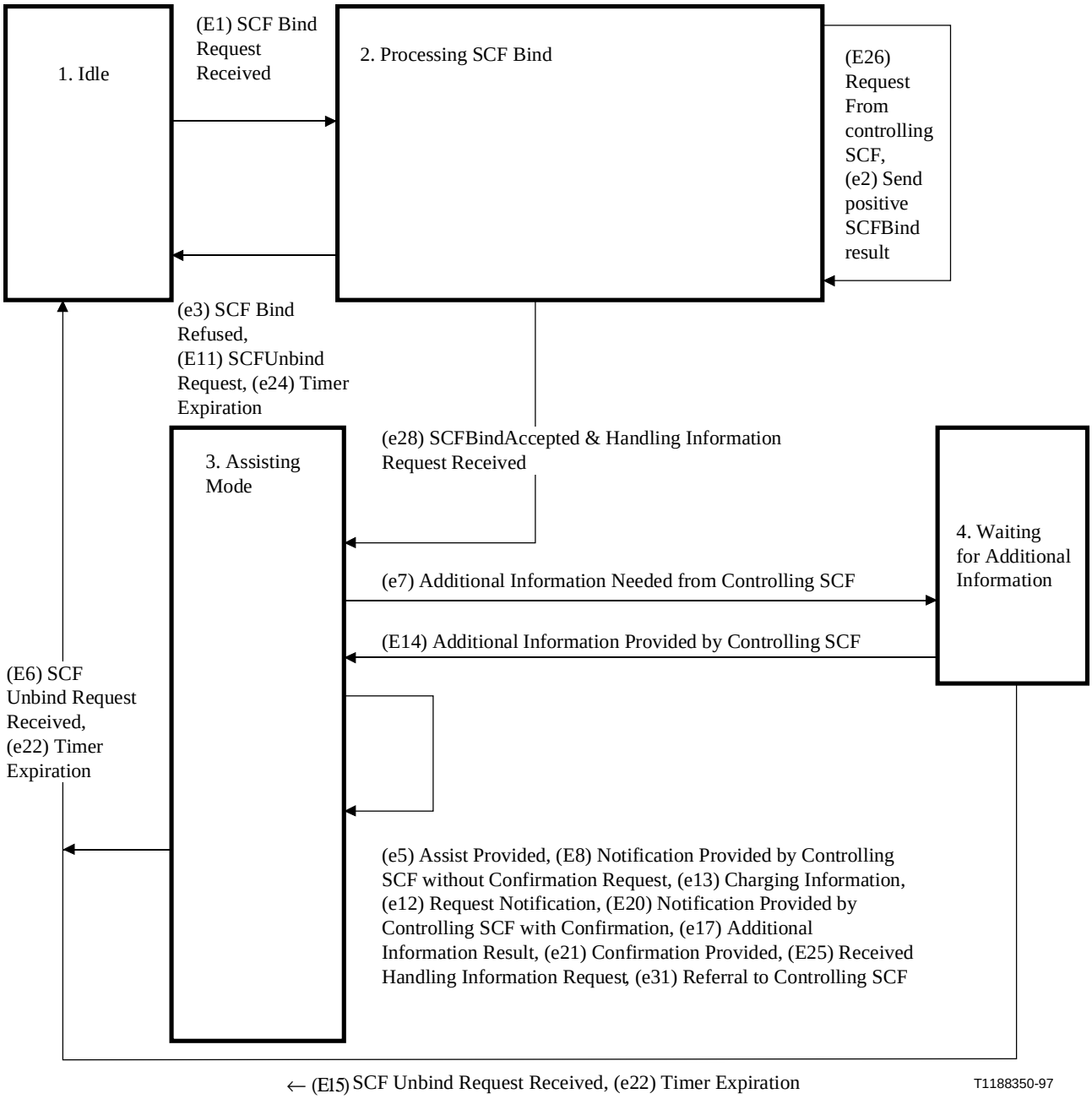
In this state, the SCF waits for information to be provided by the controlling SCF. This information should help the SCF to provide assistance to the controlling SCF. The following events are considered in this state:

- (E14) Additional_Information_Provided_by_Controlling_SCF: This is an external event caused by the reception of the response to a previously issued operation requesting further information. This event causes a transition out of this state to state 3, **Assisting mode**. This event is caused by a reception of the following operation's return result:

  – **Result of ProvideUserInformation**

– **Result of NetworkCapability**

- (E15) SCF Unbind request received : This is an external event caused by the reception of an SCFUnbind request from controlling SCF. It gives an indication from the controlling SCF that the SCF-SCF relationship needs to be ended. This event causes a transition back to state 1, **Idle**.
- (e22) Timer expiration: This is an internal event that occurs when no operation has been received from the controlling SCF for an appropriate period of time. The supporting SCF determines that the relation is no longer in existence and moves back to **Idle** state.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-20/Q.1228 – SCSM-Sup (Supporting SCF FSM)

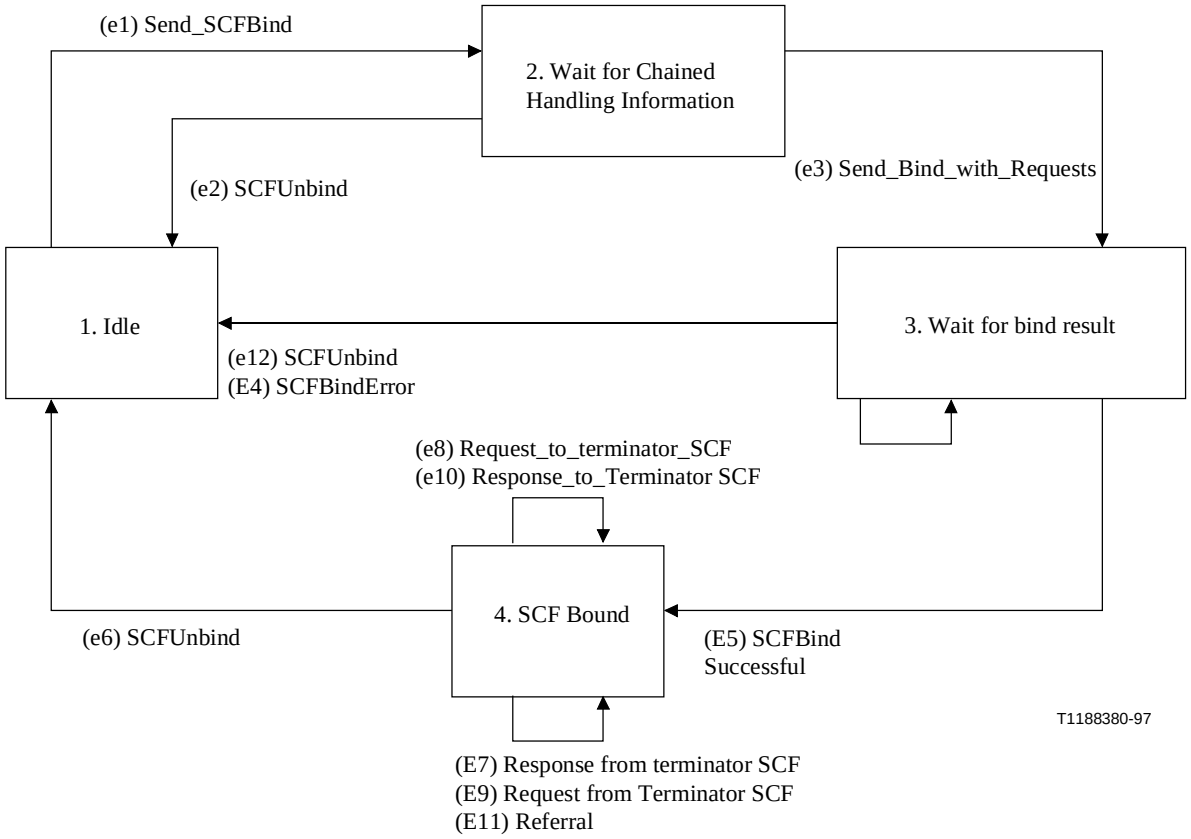
12.5.3.2.5 SCF state transition models for chaining

As for the chaining procedure, an SCF can act as a chaining initiator and as a chaining terminator. Therefore, there are two FSMs as described below.

In the following FSMs, the possibility of sending the **SCFBind** operation together with **ChainedHandlingInformationRequest** operation in one TC message is taken into consideration.

12.5.3.2.5.1 SCF state transition models for chaining initiation (SCSM-ChI)

The Finite State Machine for an SCF interacting with another SCF when acting as a chaining initiator is depicted in Figure 12-21.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-21/Q.1228 – SCSM-ChI (Chaining Initiator SCF FSM)

State 1: Idle

The only event accepted in this state is:

- (e1) Send_SCFBind: This is an internal event caused by the need to send a **SCFBind** operation to the SCSM-ChT. The **SCFBind** operations are prepared and saved. This event causes a transition out of this state to state 2, **Wait for Chained Handling Information Request**.

State 2: Wait for Chained Handling Information Request

In this state, a **SCFBind** operation has been prepared for sending and the FSM is waiting to see if a **ChainedHandlingInformationRequest** operation is to be sent with the **SCFBind** operation. The following events can occur:

- (e3) Send_Bind_with_Requests: This is an internal event caused by either the receiving of **HandlingInformationRequest** operation to be chained or an internal delimiter indicating no **HandlingInformationRequest** operation is present. It causes the **SCFBind** operations (and **ChainedHandlingInformationRequest** operation) to be sent to the SCSM-ChT. It causes a transition to state 3, **Wait for Bind Result**.
- (e2) SCFUnbind: This is an internal event, caused by the receiving of a SCFUnbind operation which requires chaining to the SCSM-ChT. This event causes a transition out of this state to state 1, **Idle**.

State 3: Wait for Bind Result

In this state, a **SCFBind** operation and an additional **ChainedHandlingInformationRequest** operation has been sent to the SCSM-ChT. The SCSM-ChT is performing the processing associated with the **SCFBind** operation (e.g. access authentication). The following events are considered in this state:

- (E4) SCFBind_Error: This is an external event caused by the failure of the **SCFBind** operation previously issued to the SCSM-ChT. This event causes a transition out of this state to state 1, **Idle**.
- (E5) SCFBind_Successful: This is an external event caused by the reception of the SCFBind confirmation for the **SCFBind** operation previously issued to the SCSM-ChT. This event causes a transition out of this state to state 4, **SCF Bound**.
- (e12) SCFUnbind: This is an internal event, caused by the receiving of a **SCFUnbind** operation which requires chaining to the SCSM-ChT. The **SCFUnbind** operation is sent to the SCSM-ChT, the SCF-SCF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**.

State 4: SCF Bound

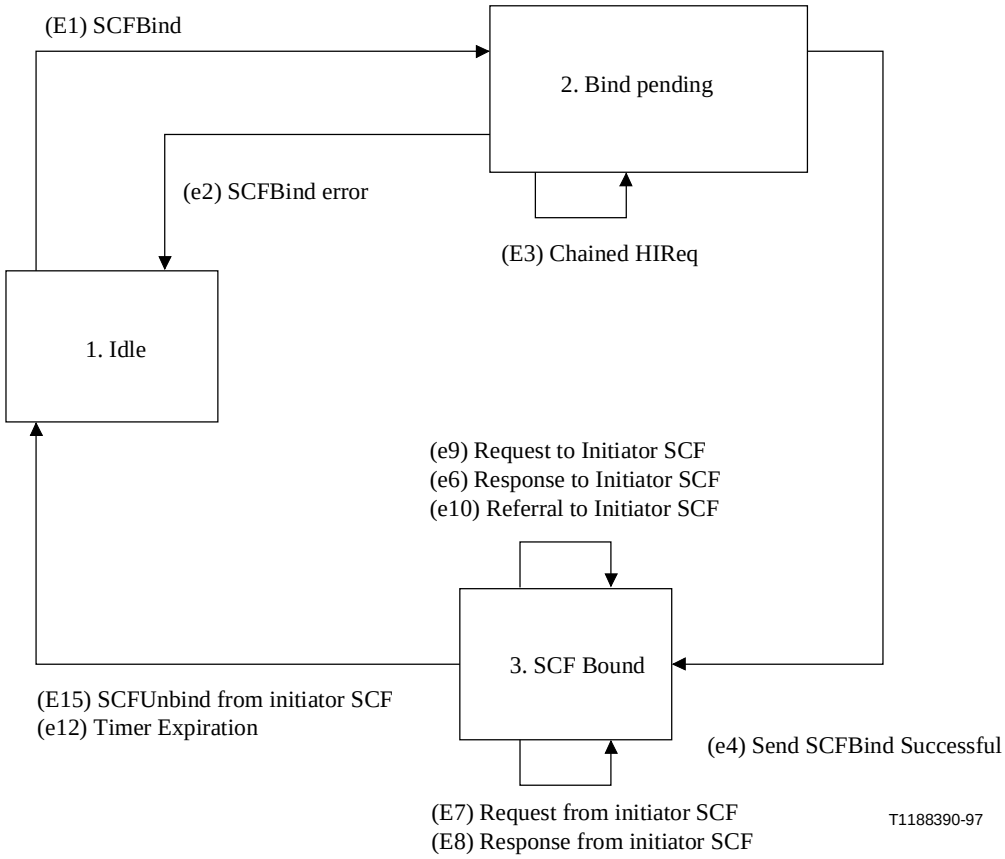
In this state, the access of the SCSM-ChI to the SCSM-ChT was authorized, chained operations can be sent to the SCSM-ChT, and results of chained operations coming from the SCSM-ChT are accepted. Thus, the SCSM-ChT will not further chain outgoing chained requests (except where the chaining initiator SCF is in another network). The following events are considered in this state:

- (e6) SCFUnbind: This is an internal event, caused by the receiving of a **SCFUnbind** operation which requires chaining to the SCSM-ChT and causes the sending of the **SCFUnbind** operation to the SCSM-ChT. The SCF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**.
- (E7) Response_from_terminator_SCF: This is an external event, caused by the reception results of the operations previously issued by the SCSM-ChI. The SCSM-ChI remains in the same state.
- (e8) Request_to_terminator_SCF: This is an internal event, caused by the sending of a chained operation to the SCSM-ChT. The SCSM-ChI remains in the same state.
- (E9) Request_from_Terminator_SCF: This is an external event caused by the reception of a chained operation from the SCSM-ChT. The SCSM-ChI remains in the same state.

- (e10) Response_to_terminator_SCF: This is an internal event caused by the sending of a result to a previously chained operation from the SCSM-ChT. The SCSM-ChT remains in the same state.
- (E11) Referral: This is an external event caused by the reception of a referral error to a **ChainedHandlingInformationRequest**. It does not cause a transition out from **SCF Bound**.

12.5.3.2.5.2 SCF state transition models for chaining termination (SCSM-ChT)

The Finite State Machine for an SCF interacting with another SCF when acting as a chaining terminator is depicted in Figure 12-22.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-22/Q.1228 – SCSM-ChT (Chaining Terminator SCF FSM)

State 1: Idle

The only event accepted in this state is:

- (E1) SCFBind: This is an external event caused by the reception of a **SCFBind** operation from an SDSM-ChI. This event causes a transition out of this state to state 2, **Bind Pending**.

State 2: Bind Pending

In this state, a SCFBind request has been received from the SCSM-ChI. The SCSM-ChT is performing the SCF access control procedures associated with the **SCFBind** operation (e.g. access authentication). The following events are considered in this state:

- (e2) SCFBind_Error: This is an internal event caused by the failure of the **SCFBind** operation previously issued from the SCSM-ChI. This event causes a transition out of this state to state 1, **Idle**, and a Bind error is returned to the SCSM-ChI.
- (E3) ChainedHandlingInformationRequest: This is an external event caused by the reception of a **ChainedHandlingInformationRequest** from the chaining initiator supporting SCF. The SCSM-ChT remains in the same state.
- (e4) SCFBind_Successful: This is an internal event caused by the successful completion of the **SCFBind** operation previously issued from the SCSM-ChI. This event causes a transition out of this state to state 3, **SCF Bound**.

State 3: SCF Bound

In this state, the access of the SCSM-ChI to the SCSM-ChT was authorized and chained operations coming from the SCSM-ChI are accepted. Besides waiting for requests from the SCSM-ChI, the SCSM-ChT can send in this state operations as responses to previously received operations, if any. The following events are considered in this state:

- (E5) SCFUnbind_from_SCF: This is an external event, caused by the reception of the **SCFUnbind** operation from the SCSM-ChI. The SCF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**.
- (e6) Response_to_initiator SCF: This is an internal event, caused by the sending of an operation to SCSM-ChI. The SCSM-ChT remains in the same state.
- (E7) Request_from_initiator SCF: This is an external event, caused by the reception of a request from the SCSM-ChI. The SCSM-ChT remains in the same state.
- (E8) Response_from_Initiator_SCF: This is an external event caused by the reception of an operation as a result to a previously chained operation sent to the SCSM-ChI. The SCSM-ChT remains in the same state.
- (e9) Request_to_initiator_SCF: This is an internal event caused by the sending of a operation to the SCSM-ChI in order to request some type of processing in the SCSM-ChI. The SCSM-ChT remains in the same state.
- (e12) Timer Expiration: This is an internal event caused by a timer expiration. It causes a transition from state ‘SCFBound' back to state 1, **Idle**.
- (e10) Referral to Initiator SCF: This is a internal event caused by the sending of a Referral error as a response to the pending **ChainedHandlingInformationRequest** to the chaining initiator supporting SCF. SCSM-ChT stays in the same state.

12.5.4 CUSF-related states (SCSM-CUSF)

Figure 12-23 shows State Diagram of the SCSM as relevant to the procedures concerning the FSM for CUSF part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

An instance of FSM for CUSF is created on reception of a DP specific instruction from the CUSF . The instance is released when the state of the instance of FSM for CUSF transits to the state, **Idle**.

The letter 'N' is added to the head of the number of each state and event in FSM for CUSF to distinguish the states and events in FSM for CUSF from those in other FSMs in the SCSM.

12.5.4.1 State N1: Idle

The following events are considered in this state:

- (Ne1) CUSF_Initiate_Control_Requested: This is an internal event caused by the service logic's need to have a new control relationship with CUSF. The FSM for CUSF requests to transmit the **InitiateAssociation** operation to the CUSF. This event causes a transition to the state N2, **Preparing CUSF Instructions**. (iv)
- (NE2) (ii) Query_from_CUSF: This is an external event, caused by a reception of one of the following operations:
  - **ActivationReceivedAndAuthorized** (for TDP-R);
  - **ComponentReceived** (for TDP-R);
  - **AssociationReleaseRequested** (for TDP-R). (i)

  This event causes a transition to state N2, **Preparing CUSF Instructions.**
- (NE3) (ii) Notification_from_CUSF: This is an external event, caused by a reception of one of the following operations:
  - **ActivationReceivedAndAuthorized** (for TDP-N);
  - **ComponentReceived** (for TDP-N);
  - **AssociationReleaseRequested** (for TDP-N). (i)

  This event causes a transition back to the same state.

12.5.4.2 State N2: Preparing CUSF Instructions

In this state, FSM for CUSF prepares appropriate instructions to the CUSF.

The following event is considered in this state:

- (Ne4) (ii) Processing_completed: This is an internal event. In this case, the SCF has completed the processing of the instructions to the CUSF. This event causes the following operation to be sent to the CUSF and a transition to state N1, **Idle**:
  - **ReleaseAssociation**

To further describe the procedures relevant to this state, the state is divided into two sub-states, which are described in the following two subclauses (this subdivision is illustrated in Figure 12-24).

12.5.4.2.1 State N2.1: Preparing CUSF Instructions

In this state, FSM for CUSF determines whether the BCUSM processing will be resumed or not, and deals with a EDP relating processing.

The following events are considered in this state:

- (Ne2.1) Event_Request: This is an internal event caused by the service logic when there is a need to send such an operation to the CUSF. It caused one or more of the **RequestReportBCUSMEvent** (iv) operations to be issued to the CUSF.

  This event causes a transition back to state N2.1, **Preparing CUSF Instructions**.
- (Ne2.2) Request_Send_Component (monitor not required): This is an internal event caused by the service logic when there is no armed EDP but a need to send **SendComponent** operation to the CUSF. It caused one or more of the **SendComponent** operation (iv) to be issued to the CUSF.

  This event causes a transition back to state N2.1, **Preparing CUSF Instructions**.

- (Ne2.3) Request_Send_Component (monitor required): This is an internal event caused by the service logic when there is(are) an(iv) armed EDP(s) and a need to send **SendComponent** operation to the CUSF. It caused one **SendComponent** operation (iv) to be issued to the CUSF.

  This event causes a transition to state N2.2, **Waiting for Notification or Request**.
- (Ne2.4) Request_Release_Association: This is an internal event caused by the service logic when it needs to release the association between the user and the network. It caused the **ReleaseAssociation** operation to be issued to the CUSF.

  This event maps into the FSM for CUSF in the SCSM event (Ne4).

12.5.4.2.2 State N2.2: Waiting for Notification or Request

In this state, FSM for CUSF waits for a notification or a request from the CUSF.

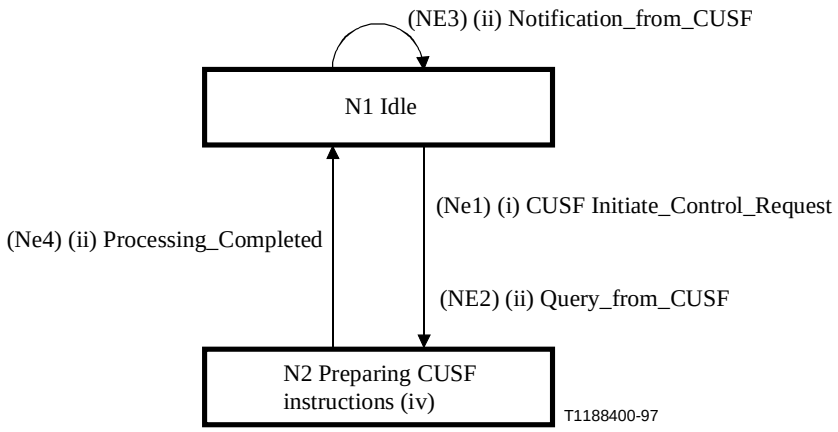
The following events are considered in this state:

- (NE2.5) EDP-R: This is an external event caused by the reception of the following operation(s):
  - **ComponentReceived** (for EDP-R); (iii)
  - **AssociationReleaseRequested** (for EDP-R). (i)

  This event causes a transition to state N2.1, **Preparing CUSF Instructions**.
- (NE2.6) Not_Last_EDP-N: This is an external event caused by the reception of the following operation(s)
  - **ComponentReceived** (for EDP-N); (iii)
  - **AssociationReleaseRequested** (for EDP-N). (i)

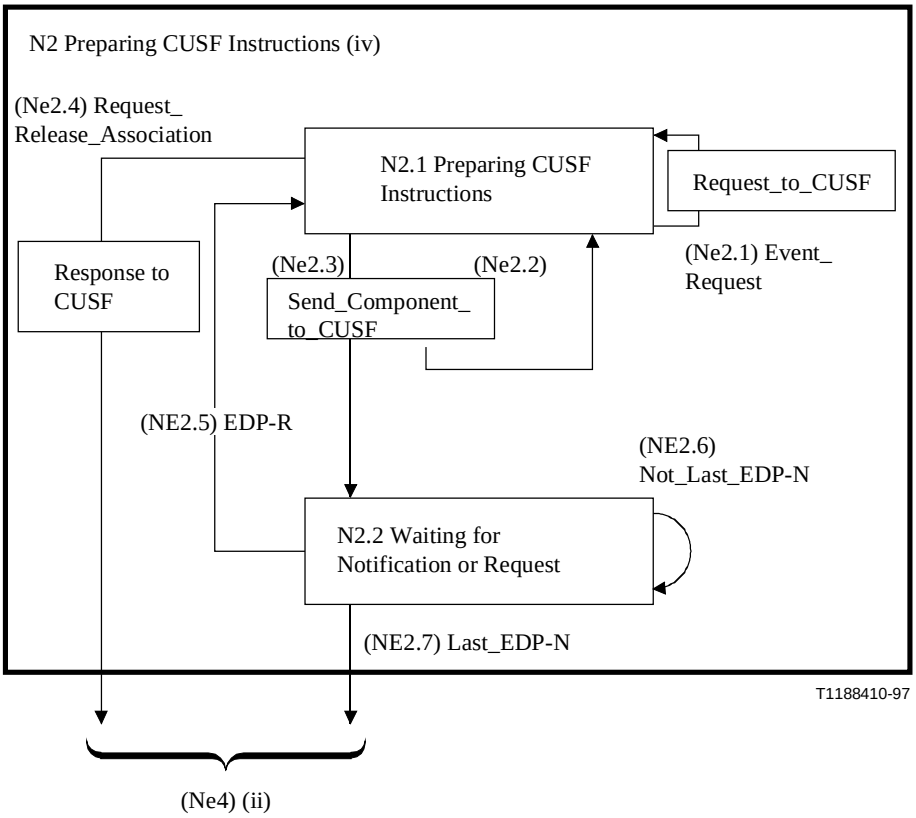
  In this case, there is still an outstanding armed EDP. This event causes a transition back to state N2.2, **Waiting for Notification or Request**.
- (NE2. 7) Last_EDP-N: This is an external event caused by the reception of the following operation(s)
  - **ComponentReceived** (for EDP-N); (iii)
  - **AssociationReleaseRequested** (for EDP-N). (i)

  This event maps into the FSM for CUSF in the SCSM event (Ne4). (ii)



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-23/Q.1228 – FSM for CUSF



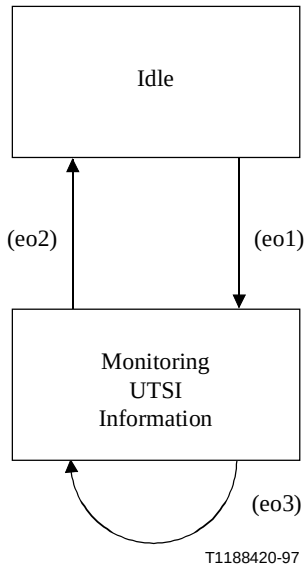
Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

NOTE – (Ne2.2) Request_Send_Component (monitor not required)
(Ne2.3) Request_Send_Component (monitor required)

Figure 12-24/Q.1228 – Sub-states of state N2

12.5.5 USI_SCF FSM

The USI_SCF FSM illustrates the SCF monitoring or not the receipt of the UTSI IE. This FSM has two states which are "**Monitoring UTSI information**" and "**Idle**" (see Figure 12-25):



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 12-25/Q.1228 – USI_SCF FSM

The USI_SCF FSM transitions are defined in the following way:

– (eo1): The SCF requests the SSF to monitor the receipt of an UTSI IE with a given *USIServiceIndicator* value.

– (eo2): The SCF is no longer interested in the receipt of an UTSI IE with the given *USIServiceIndicator* value.

– (eo3): The SCF sends an STUI IF to the User and/or receives an UTSI IE from the User with the given *USIServiceIndicator* value.

With the same operation, the SCF requests the SSF to monitor or to stop monitoring the receipt of an UTSI IE with a given *USIServiceIndicator* value.

NOTE – As an SCF controls or monitors the call, the SSF FSM is in any state except "**Idle**"; but the USI mechanism does not cause any transition in the SSF FSM.

13 SRF application entity procedures

13.1 General

This clause provides the definition of the SRF Application Entity (AE) procedures related to the SRF-SCF interface. The procedures are based on the use of Signalling System No. 7 (SS No.7); other signalling systems can be used.

Other capabilities may be supported in an implementation-dependent manner in the IP, SSP or SN.

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes TCAP (Transaction Capabilities Application Part) and one or more ASEs called TC-users. The

following subclauses define the TC-user ASE and SACF & MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

The procedure may equally be used with other message-based signalling systems supporting the application layer structures defined.

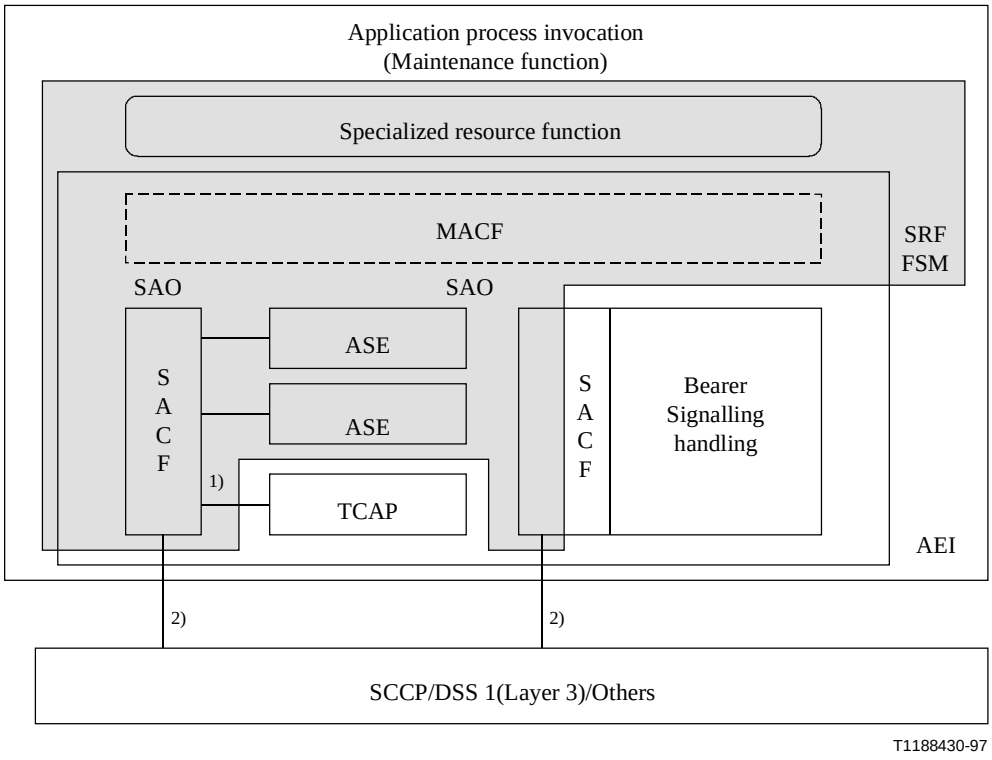
In case interpretations for the application entity procedures defined in the following differ from the detailed procedures and the rules for using TCAP services, the statements and rules contained in the detailed clauses 17 and 18 shall be followed.

Information on the SCF-External SRF communication in the relay case can be found in 3.1.2.1

13.2 Model and interfaces

The functional model of the AE-SRF is shown in Figure 13-1; the ASEs interface to TCAP (to communicate with the SCF) as well as interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 13-1.

The interfaces shown in Figure 13-1 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-Primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clauses 3 to 10.



AEI Application Entity Invocation
SRF Specialized Resource Function
FSM Finite State Machine
MACF Multiple Association Control Function
SACF Single Association Control Function
SAO Single Association Object

1) TC-primitives or Q.932-primitives.

2) N-primitives.

Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

NOTE – The SRF FSM includes several finite state machines.

Figure 13-1/Q.1228 – Functional model of SRF AE

13.3 Relationship between the SRF FSM and maintenance functions/bearer connection handling

The primitive interface between the SRF FSM and the maintenance functions is an internal interface and is not subject for standardization in IN CS-2.

The relationship between the bearer connection handling and the SRF FSM may be described as follows for the case of a call initiated by the SSF: When a call attempt is initiated by the SSF, an instance of an SRF FSM is created.

The SRF FSM handles the interaction with the SCF FSM and the SSF FSM.

The management functions related to the execution of operations received from the SCF are executed by the SRF Management Entity (SRME). The SRME interfaces the different SRF call State Models (SRSM) and the functional entity access manager (FEAM). Figure 13-2 shows the SRF FSM structure.

The model associates a Finite State Machine (FSM) with each initial interaction request from the SCF. Thus, multiple initial requests may be executed concurrently and asynchronously by the SRF, which explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SRSM objects. This entity is called the SRF Management Entity (SRME). In addition to the above tasks, the SRME maintains the dialogues with the SCF and SSF on behalf of all instances of the SCSM. In particular, the SRME;

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

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

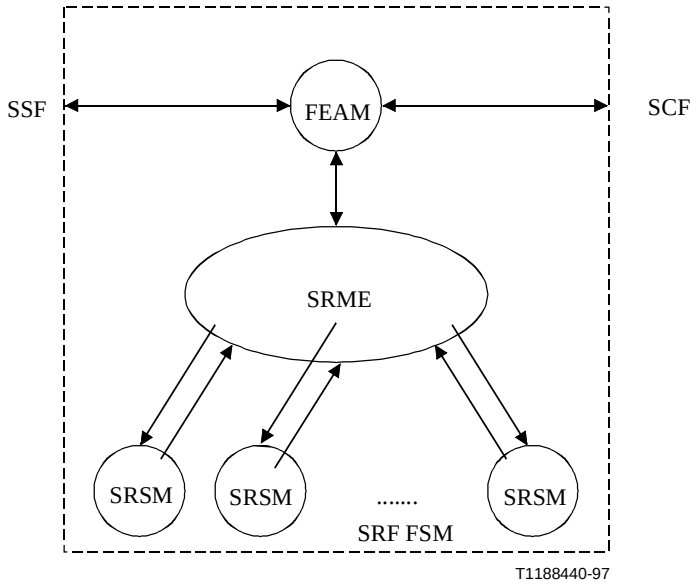
NOTE – Such a request from the SCF is executed by the SCSM when it is in its state 4.

Finally, the Functional Entity Access Manager (FEAM) relieves the SRME of low-level interface functions. The FEAM functions include:

1) establishing and maintaining the interfaces to the SSF and SCF;

2) passing (and queueing when necessary) the messages received from the SSF and SCF to the SRME; and

3) formatting, queueing (when necessary), and sending messages received from the SRME to the SSF and SCF.



FEAM Functional Entity Access Manager
SRME SRF Management Entity
SRSM SRF call State Model

Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 13-2/Q.1228 – SRF FSM structure

13.4 The SRSM

The SRSM is presented in Figure 13-3. In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. Finally, the outputs are

presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One component or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and it is processed as follows:

– The SRSM processes the operations in the order in which they are received.

– The SRSM examines subsequent operations in the sequence. When a Cancel (for PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage) operation is encountered in the sequence in state **user interaction**, it executes it immediately. In all other cases, the SRSM queues the operations and awaits an event (such an event would be the completion of the operation being executed, or reception of an external event).

– If there is an error in processing one of the operations in the sequence, the SRF FSM processes the error (see below) and discards all remaining operations in the sequence.

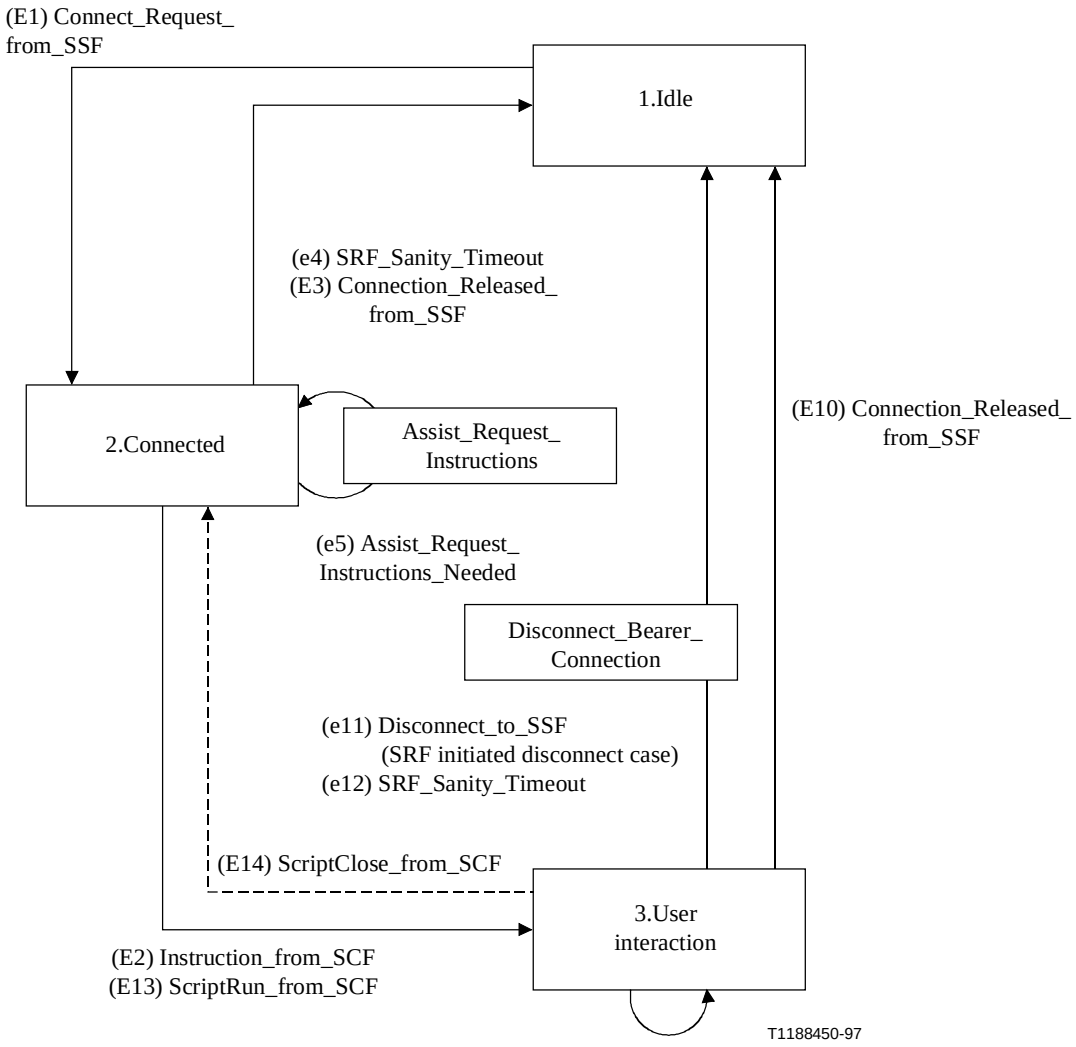
– If an operation is not understood or is out of context (i.e. it violates the SACF rules defined by the SRSM) as described above, the SRF FSM processes the error according to the rules given in 18.1.1.2 (using TC-U-REJECT or the operation error UnexpectedComponentSequence).

In any state, if there is an error in a received operation, the maintenance functions are informed. Generally, the SRSM remains in the same state in which it received the erroneous operations, however different error treatment is possible in specific cases as described in clause 16; depending on the class of the operation, the error could be reported by the SRF to the SCF using the appropriate component (see Recommendation Q.774).

In any state, if the dialogue with the SCF (direct SCF-SRF case) is terminated, then the SRSM returns to **idle** state after ensuring that all resources allocated to the dialogue have been de-allocated. The SRF shall remain connected to the SSF as long as it has PlayAnnouncement operations active or buffered. The resources allocated to the call will be de-allocated when all announcements are completed or when the SSF disconnects the bearer connection (i.e. call party release).

In any state (except **idle**), if the SSF disconnects the bearer connection to the SRF before the SRF completes the user interaction, then the SRSM clears the call and ensures that all SRF resources allocated to the call have been de-allocated. Then it transits to the **idle** state.

The SRSM has an application timer, T_{SRF}, whose purpose is to prevent excessive call suspension time. This timer is set when the SRF sends Setup Response bearer message to the SSF (SSF relay case) or the AssistRequestInstructions operation (Direct SCF-SRF case). This timer is stopped when a request is received from the SCF. The SRF may reset T_{SRF} on transmission of the SpecializedResourceReport operation, the return result for the PromptAndCollectUserInformation operation or the return result for the PromptAndReceiveMessage operation when there is no queued user interaction operation. On the expiration of T_{SRF}, the SRSM transits to the **idle** state ensuring that all SRF resources allocated to the call have been de-allocated.



(E5) Instruction_from_SCF
(E6) Cancel_from_SCF
(e7) SRF_Report_to_SCF
(e8) PlayAnnouncement/PromptAnd CollectUserInformation/
PromptAndReceiveMessage_Cancelled_to_SCF
(e9) Cancel_Error_to_SCF
(e15) ScriptEvent_to_SCF
(E16) ScriptInformation_from_SCF
(E17) ScriptRun_from_SCF

Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 13-3/Q.1228 – The SRSM

13.4.1 State 1: Idle

The **idle** state represents the condition prior to, or at the completion of, an instance of user interaction. This state is entered as a result of events E3, e4, E10, e11 and e12. It is exited as a result of event E1.

– (E1) Connect_Request_from_SSF: This event corresponds to a bearer signalling connection request message from the SSF. The details of the bearer signalling state machine related to establishing the connection are not of interest to the FSM. The SRSM goes to state "Connected";

– (E3) Connection_Released_from_SSF: This event takes place when the SRSM receives a release message from the SSF in connected state. The SRSM goes to state "Idle";

– (e4) SRF_Sanity_Timeout: This event occurs when the SRSM has been in connected state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle";

– (E10) Connection_Released_from_SSF: This event takes place when the SRSM receives a release message from the SSF in user interaction state. The SRSM goes to state "Idle";

– (e11) Disconnect_to_SSF: This event occurs when the SCF has enabled SRF initiated disconnect by:

  i) the last PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage from SCF (E2) or (E5) with the parameter disconnectFromIPForbidden. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport operation to the SCF (e7). The SRSM goes to state "Idle"; or

  ii) or by a parameter in the ScriptRun operation; and

– (e12) SRF_Sanity_Timeout: This event occurs when the SRSM has been in User interaction state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

13.4.2 State 2: Connected

This state represents the condition of the SRSM when a bearer channel has been established between a user and the SRF but the initial PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage has not yet been received (e.g. when EstablishTemporaryConnection procedures are used). The method used to provide this bearer channel is not of interest in the FSM.

– (E1) Connect_Request_from_SSF: This event corresponds to a bearer signalling connection request message from the SSF in the Idle state. The details of the bearer signalling state machine related to establishing the connection are not of interest in the SRF FSM. The SRSM goes to state "Connected".

– (E2) Instruction_from_SCF: This event takes place when the first PlayAnnouncement, PromptAndCollectUserInformation, PromptAndReceiveMessage or EraseMessage operation(s) from the SCF is received. The SRSM goes to state "User interaction".

– (E3) Connection_Released_from_SSF: This event takes place when the SRF receives a release message from the SSF. The SRSM goes to state "Idle".

– (e4) SRF_Sanity_Timeout: This event occurs when the SRSM has been connected for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

– (e5) Assist_Request_Instructions_Needed: This event occurs when the AssistRequestInstructions operation is sent from the SRSM to the SCF in the absence of a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage event (E2) initiated by the presence of a PlayAnnouncement/PromptAndCollectUserInformation/

PromptAndReceiveMessage operation concatenated with the setup request from SSF (E1) (Direct SCF-SRF case). No state change occurs as a result of this event.

– (E13) ScriptRun_from_SCF: This event takes place when the ScriptRun operation from the SCF is received. The SRSM goes to state "User Interaction".

– (E14) ScriptClose_from_SCF: This event takes place when the ScriptClose operation from the SCF is received. The SRSM goes to state "Connected".

  Note that this state transition is permitted only if the SRSM received ScriptRun operation from SCF previously.

13.4.3 State 3: User interaction

The User interaction state indicates that communication is occurring between the user and the SRF via the bearer channel established at the Connected state. This state is entered as a result of event E2. It is exited as a result of events E10, e11 and e12. Events E5, E6, e7, e8 and e9 do not cause a state change. Event E5 also represents additional PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage operations which are buffered as discussed in the procedures.

– (E2) and (E5) Instruction_from_SCF: This event takes place when an initial or subsequent PlayAnnouncement, PromptAndCollectUserInformation, or PromptAndReceiveMessage operation(s) from the SCF is received. The SRSM goes to state "User interaction" on the first (E2). The SRSM remains in state "User interaction" for subsequent (E5)s.

– (E6) Cancel_from_SCF (for PlayAnnouncement/PromptAndCollectUserInformation /PromptAndReceiveMessage): This event takes place when the corresponding PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage operation is received from the SCF. The indicated interaction is terminated if it is presently running, otherwise it is deleted from the buffer. The SRSM remains in state "User interaction".

– (e7) SRF_Report_to_SCF: This event takes place when a SpecializedResourceReport, a return result for PromptAndCollectUserInformation or a return result for PromptAndReceiveMessage operation is sent to the SCF. The SRSM remains in state "User interaction".

– (e8) PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage_Cancelled_to_SCF: This event takes place when the PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage error caused by the Cancel (for PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage) operation is sent to the SCF. This event represents the successful cancellation of an active or buffered PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message operation. The SRSM remains in state "User interaction".

– (e9) Cancel_Error_to_SCF: This event takes place when the cancel error (for PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage) is sent to the SCF. This event represents the unsuccessful cancellation of a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. The SRSM remains in state "User interaction".

– (E10) Connection_Released_from_SSF: This event takes place when the SRSM receives a release message from the SSF. The SRSM goes to state "Idle".

– (e11) Disconnect_to_SSF: This event occurs when the SCF has enabled SRF initiated disconnect by:

  • the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message from SCF (E2) or (E5). The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport operation, a return result for PromptAndCollectUserInformation or a return result for PromptAndReceiveMessage operation to the SCF. The SRSM goes to state "Idle"; or

  • by a parameter in the ScriptRun operation.

– (e12) SRF_Sanity_Timeout: This event occurs when the SRSM has been in user interaction state for a network-operator-defined-period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

– (E14) ScriptClose_from_SCF: This event takes place when the ScriptClose operation from the SCF is received. The SRSM goes to state "Connected".

  Note that this state transition is permitted only if the SRSM received ScriptRun operation from SCF previously.

– (e15) ScriptEvent_to_SCF: This event takes place when a partial result is sent from a SRF to the SCF in case the SRF needs additional information or when the final Result of the User Interaction script execution is sent from the SRF to the SCF.The SRSM remains in the state "User Interaction".

– (E16) ScriptInformation_from_SCF: This event takes place when the ScriptInformation operation from the SCF is received. The SRSM remains in the state "User Interaction".

– (E17) ScriptRun_from_SCF: This event takes place when the ScriptRun operation from the SCF is received. The SRSM remains in state "User Interaction". It follows a PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage which is terminated. No other User Interaction Script shall be already active for the call.

  Note that a subsequent ScriptRun operation from the SCF is not permitted in this state.

In addition to these explicitly marked transitions, failure of a user-SRF bearer connection will cause the SRSM to transit to Idle from any state. These transitions are not shown on Figure 13-3 for the purpose of visual clarity.

13.5 Example SRF control procedures

This subclause provides a detailed description of the SRF procedures. Arrow diagrams are used for the description of the connect, interaction with the end user, and disconnect stages.

The SRF control procedures are based on various physical allocation patterns of SRF. The various control procedures are described in this subclause in accordance with the example physical scenarios of protocol architecture in 3.1.

The service assist and hand-off procedures based on the physical scenarios are also described in this subclause as examples.

Note that, through this subclause, bearer connection control signalling messages are used for explanatory purpose, and are not subject for standardization in this Recommendation. The terms used for bearer connection control signalling messages only represent the functional meaning.

13.5.1 SRF connect procedures

13.5.1.1 SRF connect physical procedures

Several procedures are required for different physical scenarios. The cases to be covered are described below and illustrated in Figure 13-4.

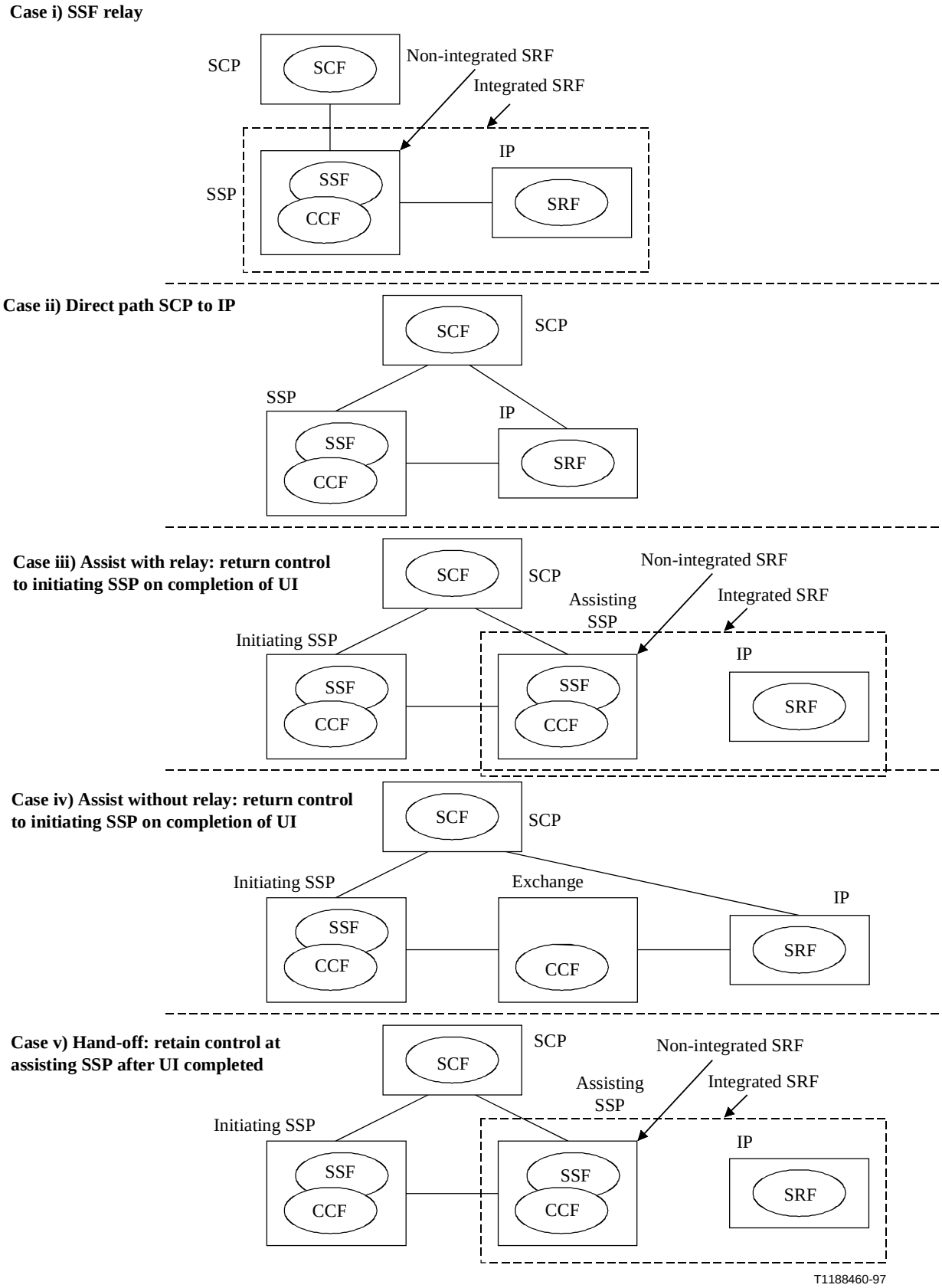


Figure 13-4/Q.1228 – Physical scenarios

i) the IP is integrated into the SSP, or attached to the SSP, possibly via a local exchange, that is interacting with the SCP but the SCP's operations to the IP are relayed via the SSP which performs any needed protocol conversion;

ii) the IP is directly attached to the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved;

iii) the IP is integrated into another SSP, or directly attached to another SSP, than the one that is interacting with the SCP but the SCP's operations to the IP are relayed via the second SSP (called the "Assist" method), and on completion of the user interaction, control is returned to the first SSP;

iv) the IP is directly attached to a node other than the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved (called the "Assist" method, but with a variation on the physical connectivity of the entities involved), and on completion of the user interaction, control is returned to the first SSP; and

v) the IP is attached to another SSP and on completion of the user interaction, control of the call is retained at that SSP (called the "Hand-off" approach).

In each of the above cases, the operations between the SCP and the SSP may be SS No. 7 TCAP-based; the messaging between the SSP and the IP when the SSP does relaying may be DSS 1 using the facility IE (in this case, the SSP would have to do protocol conversion from SS No. 7 TCAP to DSS 1 facility IE for the operations and responses it relayed between the SCP and the IP); the direct messaging between the SCP and the IP may be SS No. 7 TCAP-based; and bearer control signalling may be any system.

Each of the scenarios will now be examined using arrow diagrams.

NOTE to Figures 13-5 to 13-9 – Black lines indicate INAP operations, grey lines indicate DSS 1 operations.

Case i) is illustrated in Figure 13-5. Note that for the integrated IP/SSP, the internal activities of the node can still be modelled in this way, but the details of how this is achieved are left to the implementor. This approach makes it unnecessary for the SCP to distinguish between integrated and external but directly connected IPs. See also a note on the possibility of concatenating the first user interaction operation with the ConnectToResource operation discussed in the subclause on user interaction below. The establishment of the SCF-SRF relationship in this case is implicit.

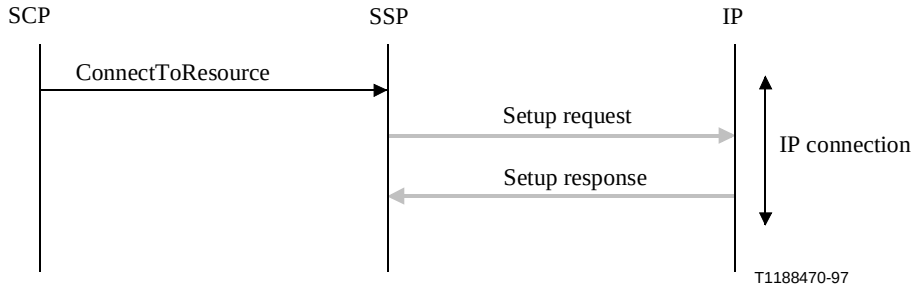


Figure 13-5/Q.1228 – Connection to integrated or external IP with SSP relay of IP operations

Case ii) requires that the IP indicate to the SCP that it is ready to receive operations (see Figure 13-6). The establishment of the SCF-SRF relationship is explicit. Note that it is necessary to convey a correlation ID to ensure that the transaction established between the SCP and the IP can be correlated to the bearer connection setup as a result of the SCP's preceding operation to the SSP.

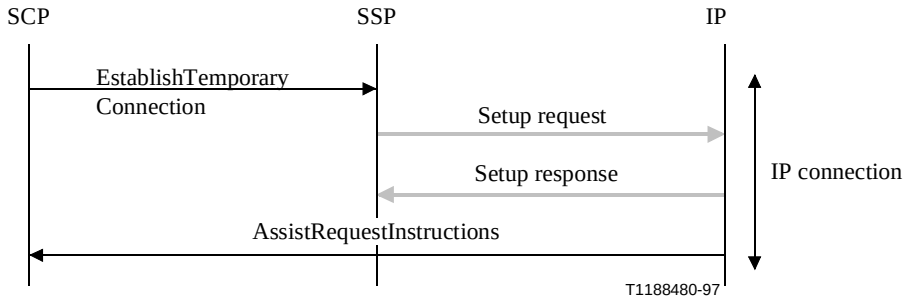


Figure 13-6/Q.1228 – Connection to IP with direct link to SCP, IP initiates interaction with SCP

Case iii) requires that a transaction be opened with the assisting SSP so that it may relay operations from the SCP to the IP (integrated or external). Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP. After the AssistRequestInstructions are received by the SCP, the procedures are the same as case i). Figure 13-7 illustrates the preamble involved.

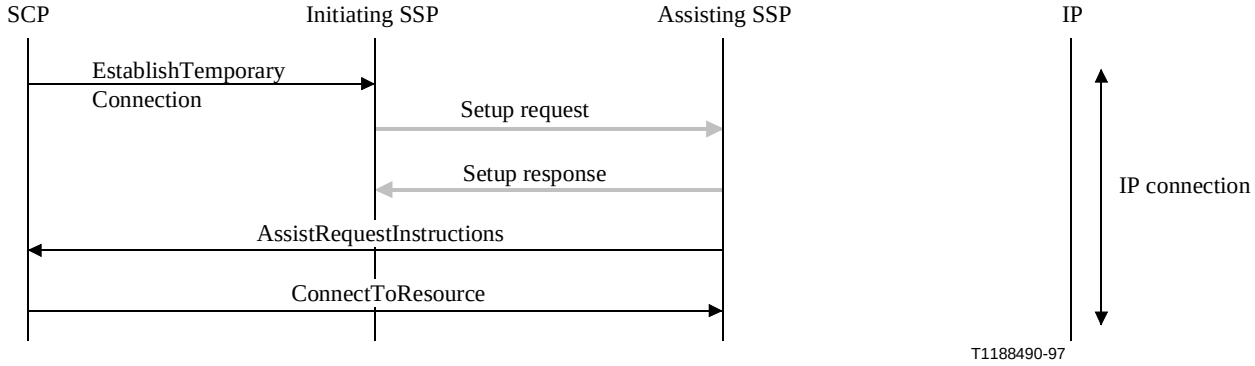


Figure 13-7/Q.1228 – Preamble for assist case with integrated IP or external IP and SSP relay of SCP-IP messages

Case iv) does not require the establishment of a second transaction from the assisting exchange, hence it need not be an SSP. This then becomes a preamble to the procedure shown in Figure 13-6 as shown in Figure 13-8.

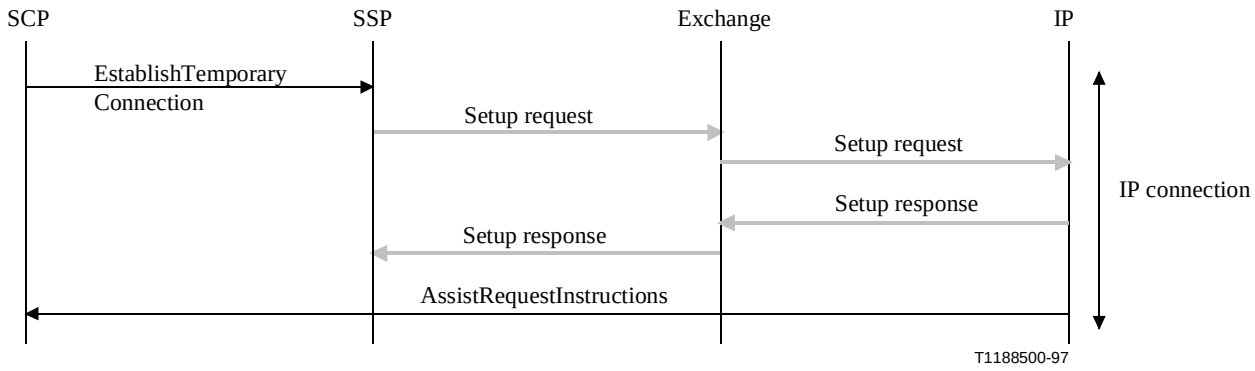


Figure 13-8/Q.1228 – Preamble for assist case with external IP and direct SCP-IP messaging

Case v) merely requires the sending of an operation to the first SSP to route the call to the handed-off SSP, and then Figure 13-5 applies at handed-off SSP. This is shown in Figure 13-9. Note that the activity at handed-off SSP represents a new interaction with the SCP and "AssistRequestInstructions" is used. Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP.

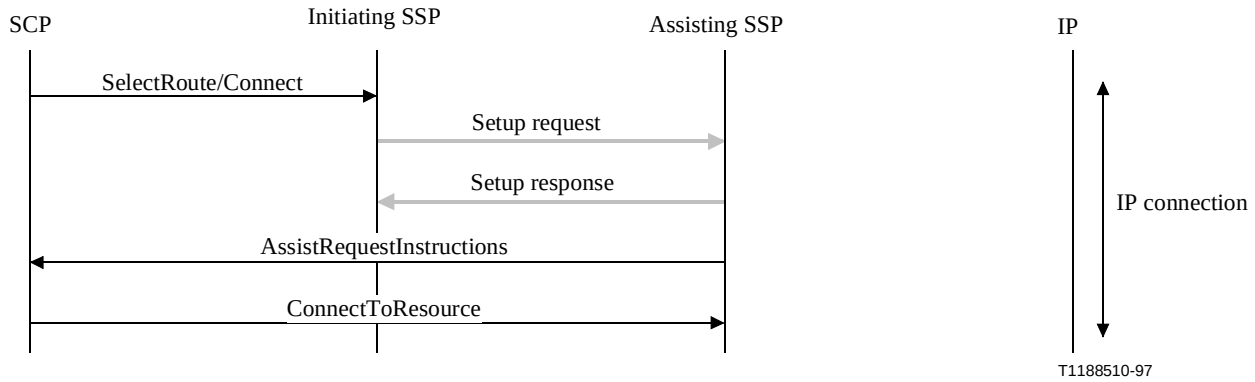


Figure 13-9/Q.1228 – Preamble for hand-off case

13.5.2 SRF end user interaction procedures

The end user interaction procedures allow:

– the sending of one or multiple messages to the end user by using the PlayAnnouncement operations;

– a dialogue with the end user by using one or a sequence of PromptAndCollectUserInformation operations;

– a dialogue with the end user by using one or a sequence of PromptAndReceiveMessage operations;

– a combination of the above; and

– cancellation of a PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReciveMessage operations by using a generic cancel operation.

13.5.2.1 Play announcement/prompt & collect user information/prompt & receive message (PA/P&C/P&R)

There are only two physical scenarios for user interaction:

i) the SSP relays the operations from the SCP to the IP and the responses from the IP to the SCP (SSF relay case); and

ii) The operations from the SCP to the IP and the responses from the IP are sent directly between the SCP and the IP without involving the SSP (direct SCF-SRF case).

Case i) is illustrated in Figure 13-10 below.

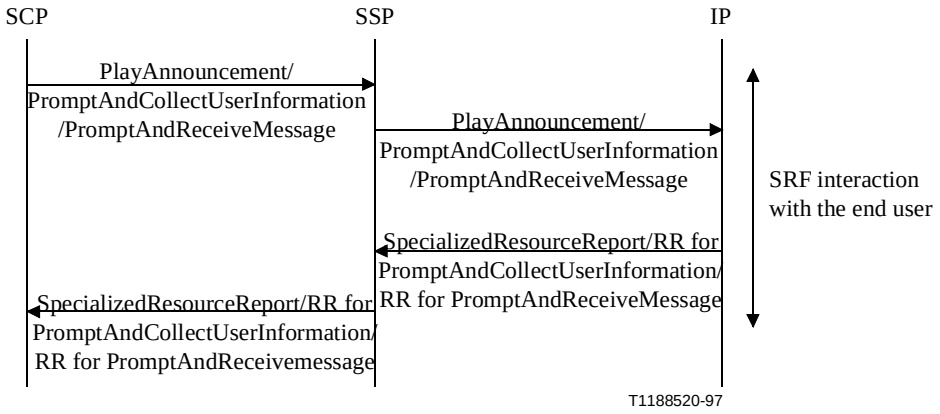


Figure 13-10/Q.1228 – SSP relay of user interaction operations and responses

Case ii) is illustrated in Figure 13-11 below.

It is also necessary to consider the capability of SS No. 7 TCAP to concatenate several Invoke PDUs in one message. This capability allows, for the scenario in Figure 13-5, the ConnectToResource and the first PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage to be carried in one message. This has some advantages in this physical scenario, such as reduced numbers of messages, and possibly better end-user perceived performance.

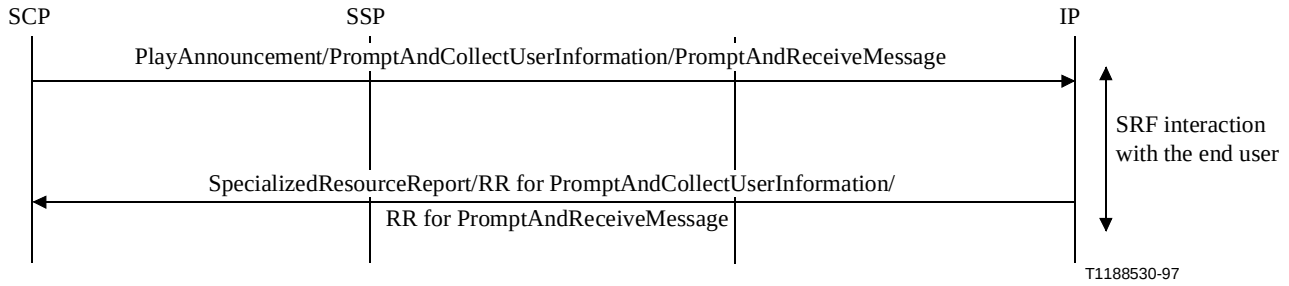


Figure 13-11/Q.1228 – Direct SCF-SRF of user interaction operations and responses

13.5.3 SRF disconnection procedures

The disconnection procedures are controlled by the SCF and the procedure used is selected based on the needs of the service being executed. The bearer disconnection procedure selected by the SCF is to either allow the SRF to disconnect on completion of user interaction, or to have the SCF explicitly order the SSF to disconnect.

SRF disconnect does not cause disconnection by the SSF/CCF back to the end user terminal unless the transaction with the SCF has been terminated, indicating the user interaction completed the call. The SSF/CCF recognizes that a connection to an SRF is involved because the operations from the SCF for this purpose are distinct from the operations that would be used to route the call towards a destination. There is no impact on bearer signalling state machines as a result of this since incoming and outgoing bearer signalling events are not simply transferred to each other, but rather are absorbed in call processing, and regenerated as needed by call processing. Therefore, to achieve the desired functionality, call processing need simply choose not to regenerate the disconnect in the backward direction. Figure 13-12 illustrates this concept.

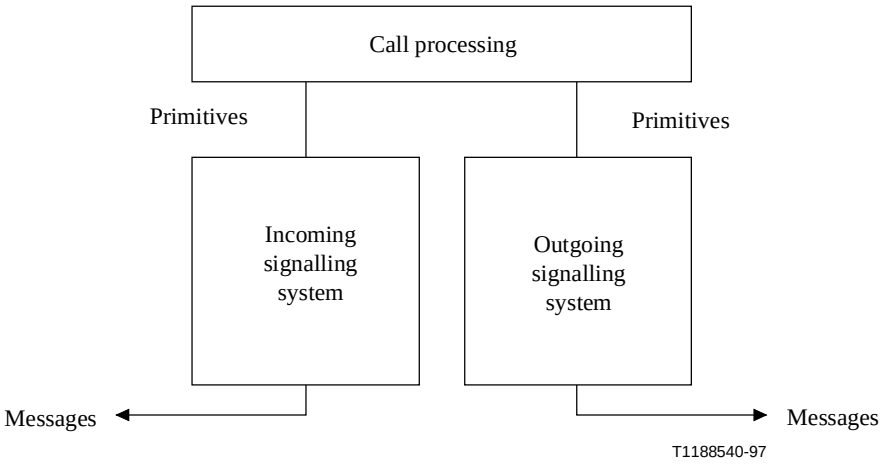


Figure 13-12/Q.1228 – Relationship of incoming and outgoing signalling systems to call processing

As for the SRF connection procedures, the SRF disconnection is affected by the physical network configuration.

In order to simplify the interface between the SCF and the SRF, a number of assumptions are made. The assumptions, and the resulting rules, result in unambiguous procedures from both the SCF and the SRF points of view. The rules, presented below, refer to the SRF originated disconnect, or "SRF Initiated Disconnect", and to the SCF originated disconnect, or "SCF Initiated Disconnect". While other scenarios are possible, they are not included because they either duplicate the functionality presented below or they otherwise do not add value from a service perspective.

1) If a series of PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage operations are to be executed by the same SRF, then SRF disconnect is inhibited for all but the last and may be inhibited on the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage. When a subsequent PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage is received, it is buffered until the completion of any preceding PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage.

2) A generic cancel operation terminates the indicated PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage if it is being executed by the SRF, but does not disconnect the SRF. If the cancel operation is for a buffered PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage, that PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage is discarded, but the current and any buffered PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage are executed. An SRF interacts with one user only and therefore cancelling a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage only affects the user to which the SRF is connected.

3) The SCF must either explicitly order "Disconnect" or enable SRF initiated disconnect at the end of the PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage. An SRF left connected without a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage to execute may autonomously disconnect if it has not received any PlayAnnouncement/PromptAndCollectUserInformation/PromptAnd ReceiveMessage operations within a defined time limit. This could occur, for example, after an EstablishTemporaryConnection which is not followed within a reasonable time period with a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. This sanity timing value will depend on the nature of the interaction the SRF supports and should be selected by the network operator accordingly.

4) When SRF initiated disconnect is enabled in a PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage, then the SRF must disconnect on completion of the user interaction.

5) When SRF initiated disconnect is not enabled, the SCF must ask the SRF to inform it of the completion of the user interaction using the SpecializedResourceReport operation for "announcement complete", using the return result for the PromptAndCollectUserInformation operation or using the return result for the PromptAndReceiveMessage operation.

6) If the user disconnects, the SRF is disconnected and the SSF releases resources and handles the transaction between the SSF and the SCF as specified in Recommendation Q.1224 and in this Recommendation. The SRF discards any buffered operations and returns its resources to idle. The relationship with the SCF is terminated.

7) When the SCF explicitly orders the SSF to disconnect by "DisconnectForwardConnection" operation, the SSF releases the bearer connection to the SRF, and returns to the "waiting for instructions" state. No operation reporting SRF disconnect from the SSF to the SCF is required.

13.5.3.1 SRF initiated disconnect

The SRF disconnect procedure is illustrated in Figure 13-13. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. When the SRF receives a PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The SSF/CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The SSF returns to the "waiting for instructions" state and executes any buffered operations. In the hand-off case, the SSP shown in Figure 13-13 is the "handed-off" SSP.

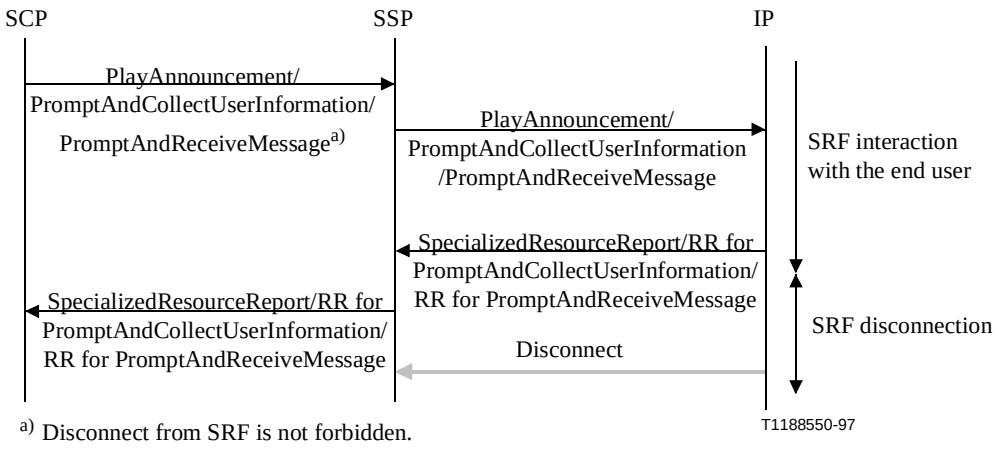


Figure 13-13/Q.1228 – SCF disconnect for local, embedded and hand-off scenarios

For the assisting SSF case, the SRF initiated disconnect procedures are not used because the assisting SSF remains in the "waiting for instructions" state and does not propagate the disconnection of the bearer connection to the Initiating SSF. The SCF initiated disconnect procedures described in the following subclause are used for the assisting SSF case.

For the direct SCF-SRF case, the procedures also work in the same manner. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage operation. When the SRF receives a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The Initiating SSF/CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The Initiating SSF returns to the "waiting for instructions" state and executes any buffered operations.

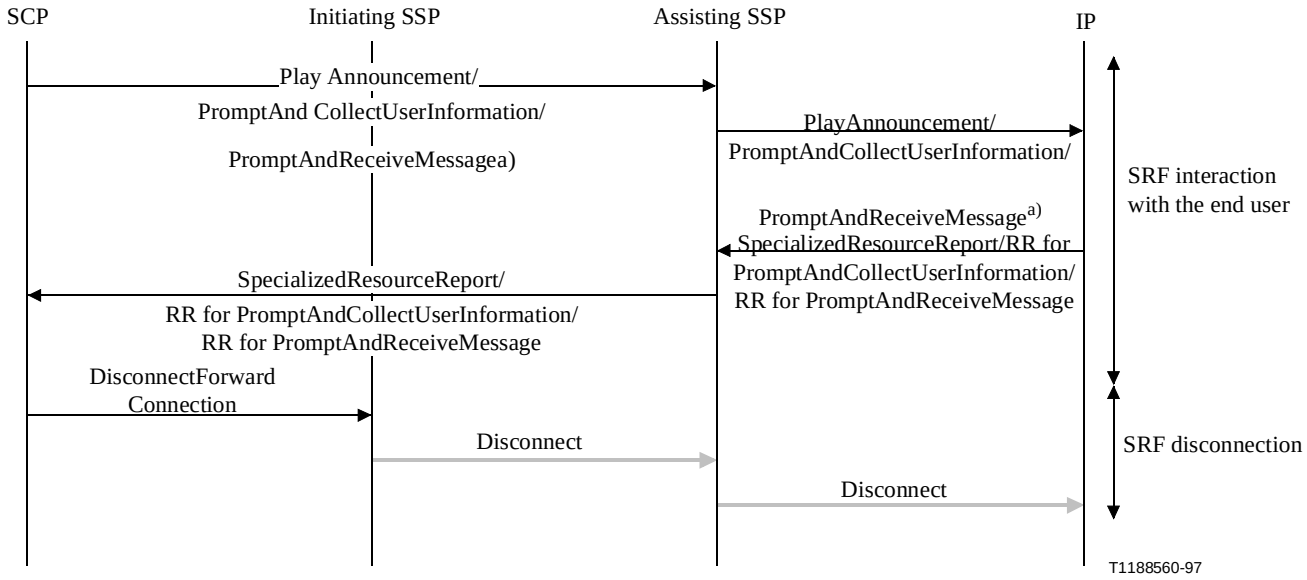
13.5.3.2 SCF Initiated Disconnect

The SCF initiated disconnect procedure is illustrated in Figure 13-14. Bearer messages are shown in gray. The figure shows only the assisting SSF case, and the direct SCF-SRF case is not shown. To initiate the SCF initiated disconnection of the SRF, the SCF must request and receive a reply to the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message operation requested. The SpecializedResourceReport operation contains an "announcement complete" and return result for PromptAndCollectUserInformation contains "collected information."

The SCF initiated disconnect uses an operation called DisconnectForwardConnection. Once the DisconnectForwardConnection is received by the SSF, it will initiate a "release of bearer channel connection" between the physical entities containing the SSF and SRF, using applicable bearer control signalling. Since the SCF (which initiates the disconnect), the SSF (which instructs bearer signalling to disconnect) and the SRF (which receives disconnect notification via bearer signalling) are aware that disconnect is occurring, they are synchronized. Therefore, a "pre-arranged" end may be used to close the transaction. This does not preclude the use of explicit end messages for this purpose.

For assisting SSF case, the initiating SSP, on receipt of the DisconnectForwardConnection from the SCP, disconnects forward to the assisting SSP, and this disconnection is propagated to the IP. The initiating SSP, knowing that the forward connection was initiated as the result of an

EstablishTemporaryConnection, does not disconnect back to the user but returns to the "waiting for instructions" state.



[a)] Disconnect from SRF is forbidden.

Figure 13-14/Q.1228 – SCF initiated disconnect for assist scenario

13.5.4 Examples illustrating Complete User Interaction Sequences

The following figures and their accompanying tables provide examples of complete sequences of user interaction operations covering the three stages:

– Connect the SRF and the end user (bearer connection) and establish the SCF-SRF relationship.

– Interact with the end user.

– Disconnect the SRF and the end user (bearer connection) and terminate the SCF-SRF relationship.

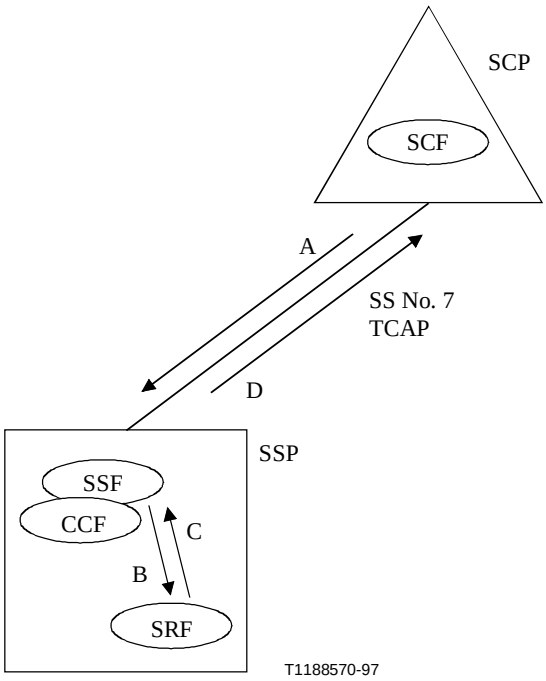


Figure 13-15/Q.1228 – SSP with integrated SRF

In Figure 13-15, the SSP with an integrated (or embedded) SRF, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA /P&C/P&R	ConnectToResource; PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMesssage Setup; PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage	A B
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage	A then B C then D
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage Disconnect	C then D C (intra-SSP bearer control)
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage DisconnectForwardConnection Disconnect	C then D A B (intra-SSP bearer control)

A simple extension to this integrated case is the configuration where the SRF is located in an intelligent peripheral locally attached to the SSP. The SCP-IP operations are relayed via the SSF in the SSP. This is depicted in Figure 13-16.

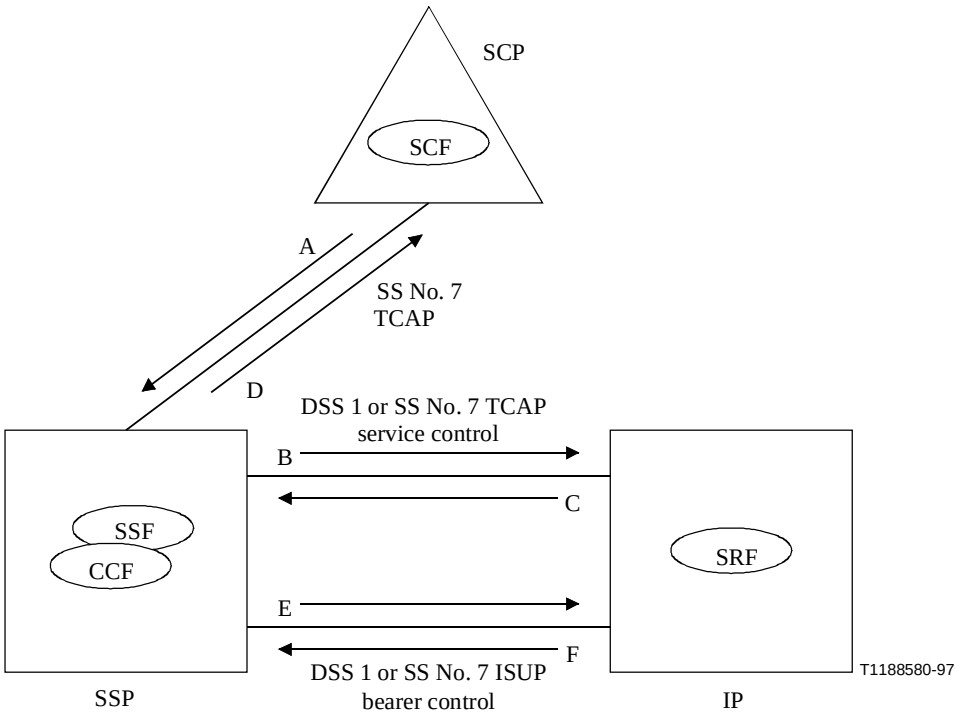


Figure 13-16/Q.1228 – SSP relays messages between SCP and IP

The procedural scenarios for this relay SSF with an IP (Figure 13-16) can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C/P&R	ConnectToResource; PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage *If DSS 1 used:* Setup; PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage *If SS No. 7 used:* IAM PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage	A E and B (Facility IE) E B
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage	A then B C then D

Procedure name	Operations	Protocol flows
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C then D F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C then D A E E

In some cases, the IP may have an SS No. 7 or other interface to the controlling SCP. This case is shown in Figure 13-17. Note that the SCP must correlate two transactions to coordinate the activities.

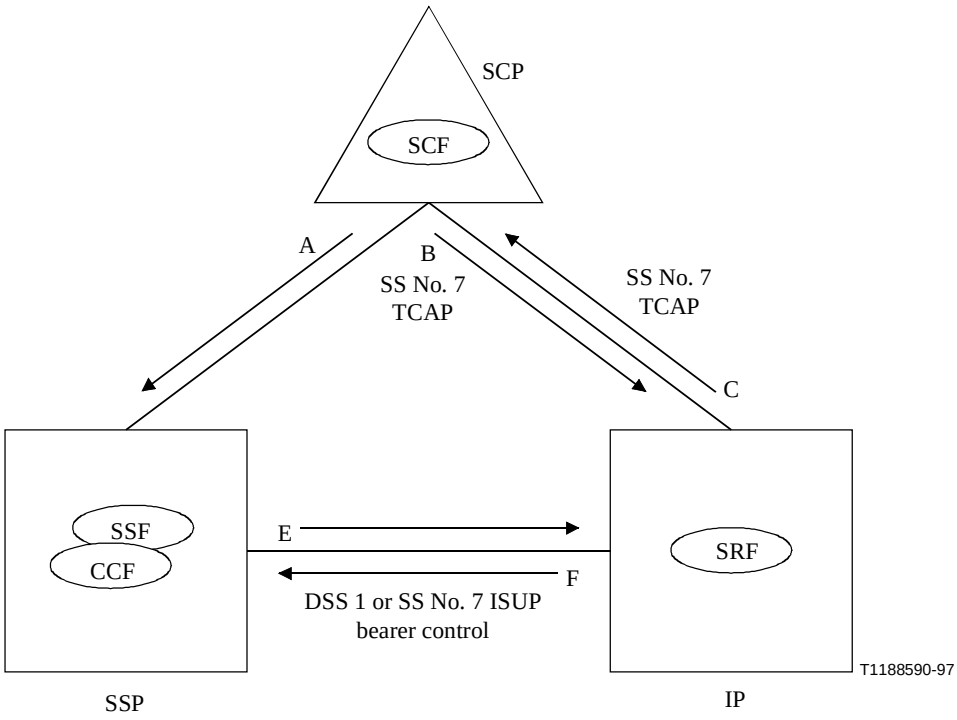


Figure 13-17/Q.1228 – Direct SCP-IP information transfer

In Figure 13-17, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource	EstablishTemporaryConnection *If DSS 1 used:* Setup AssistRequestInstructions *If SS No. 7 used:* IAM AssistRequestInstructions	A E C E C
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage	B C
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C A E E

The assisting SSF scenario involves straightforward procedural extensions to the basic cases shown above. One mapping of the assisting SSF case is shown in Figure 13-18. In this case, SRF initiated disconnect cannot be used. Other physical mappings can be derived as described in the text following the figure and its accompanying table.

Note that the integrated SRF and SSF relay case requires a transaction between the SCP and the assisting SSP (Figure 13-18) but the SCP direct case does not since the transaction is directly between the SCP and the IP connected to the remote exchange. In the latter case, any transit exchanges, including the one the IP (SRF) is connected to, are transparent to the procedures.

Note also that the SCP must again correlate two transactions.

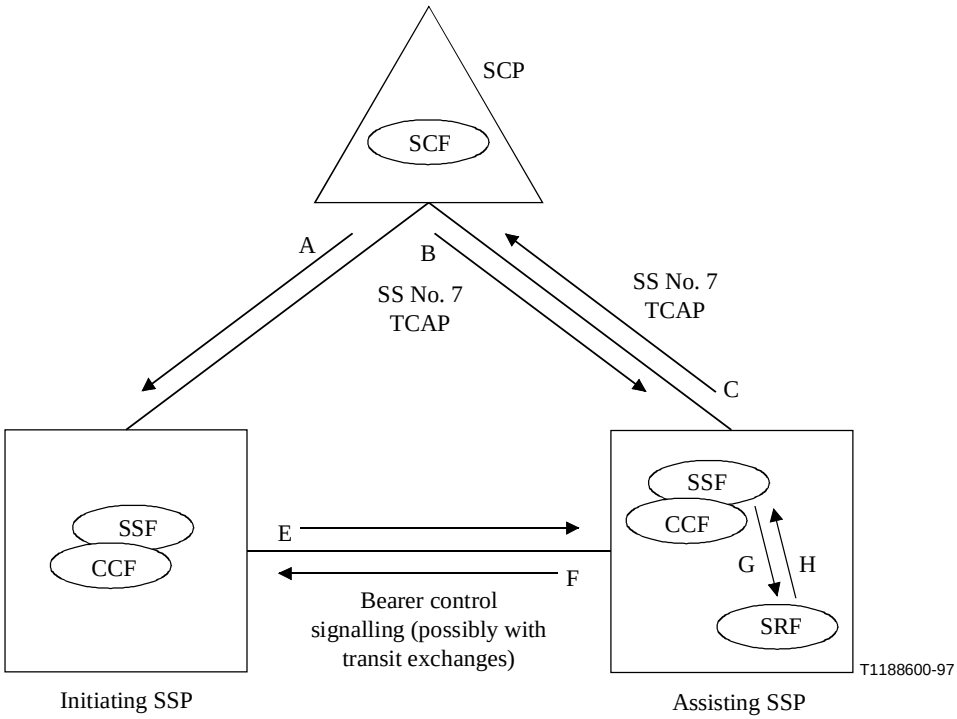


Figure 13-18/Q.1228 – SSP assist(relay SSP)

In Figure 13-18, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Assist preamble	EstablishTemporaryConnection	A
	If DSS 1 used:	
	Setup	E
	AssistRequestInstructions	C
	ConnectToResource	B
	Setup	G
	ResetTimer	A
	If SS No. 7 used:	
	IAM AssistRequestInstructions	E
	ConnectToResource	C
	Setup	B
	ResetTimer	G
		A
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/ PromptAndReceiveMessage	B then G
	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage	H then C

Procedure name	Operations	Protocol flows
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/ RR for PromptAndReceiveMessage DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	H then C A E and G (intra-SSP bearer ctrl) E and G (intra-SSP bearer ctrl)

Note that the assisting SSP case shown in Figure 13-18 can be generalized to cover both the case where the SRF is embedded in assisting SSP (as shown), and the case where the SRF is locally connected to assisting SSP. In this latter case, the SRF communication (protocol flows B, C, G and H) would conform to the physical scenario shown in Figure 13-16.

The service hand-off scenario can similarly be viewed as a sequence consisting of an IN service to route a call from one SSP to another, followed by any one of the previously described physical user interaction scenarios. For describing this scenario, Figure 13-18 can be used also.

13.5.4.1 Message sequences for service assist

This subclause provides additional details on the message sequences for the service assist procedure in Figure 13-18:

1) *Protocol Flow A* – The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing will continue from the initiating SSP after the remote resources have been used (e.g. the call will be completed to a destination address after information is collected from the calling party). An EstablishTemporaryConnection operation containing the address of the assisting SSP (for routing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP. The EstablishTemporaryConnection is used instead of a regular Connect operation because of the nature of the connection to the assisting SSP. The initiating SSP must be aware that the SCP will ask it to continue in the processing of the call at some point in the future.

   NOTE 1 – The ScfID and CorrelationID may be included in the routing address of the assisting SSP.

2) *Protocol Flow E* – The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS7 information elements (e.g. routing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

3) *Protocol Flow C* – The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions.

4) *Protocol Flow B* – The SCP sends instructions to the assisting SSP based on service logic control.

5) *Protocol Flow A* – The SCP may need to generate reset timer events to the initiating SSP so that it does not time out the call.

   NOTE 2 – The usage of ResetTimer operation is optional.

6) *Protocol Flow A* – When resource functions have been completed, a DisconnectForwardConnection operation is sent to the initiating SSP. This indicates, that the temporary connection to the assisting SSP has to be disconnected.

NOTE 3 – A DisconnectForwardConnection operation followed by a ConnectToResource may be sent to the assisting SSP to access several resources in the assisting case.

7) *Protocol Flow E* – The initiating SSP sends a message via bearer control signalling to the assisting SSP to close the "assist" transaction.

8) The call control returns to the initiating SSP.

13.5.4.2 Message sequences for hand-off

This subclause outlines message sequences for the hand-off procedure using the protocol flows shown in Figure 13-18:

1) *Protocol Flow A* – The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing need not continue from the initiating SSP after the remote resources have been used (e.g. a terminating announcement will be played). A Connect operation containing the address of the assisting SSP (for routing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP.

NOTE – The ScfID and CorrelationID may be included in the routing address of the assisting SSP.

2) *Protocol Flow E* – The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS7 information elements (e.g. routing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

3) *Protocol Flow C* – The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions. The AssistRequestInstructions is used instead of a regular request instruction (InitialDP or DP-specific operation) because the SCP must associate the AssistRequestInstructions from the assisting SSP/IP with an already active dialogue the SCP has with another SSP.

4) *Protocol Flow B* – The SCP sends instructions to the assisting SSP based on service logic control.

5) The call control remains at the assisting SSP.

The same service assist and hand-off procedures can be reused for a direct link to an IP in this and future capability sets.

14 SDF application entity procedures

14.1 General

This clause provides the definition of the SDF Application Entity (AE) procedures related to the SDF-SCF and SDF-SDF interfaces. The procedures are based on the use of Signalling System No. 7 (SS No. 7).

Other capabilities may be supported in an implementation-dependent manner in the SCP, SDP, SSP, AD or SN.

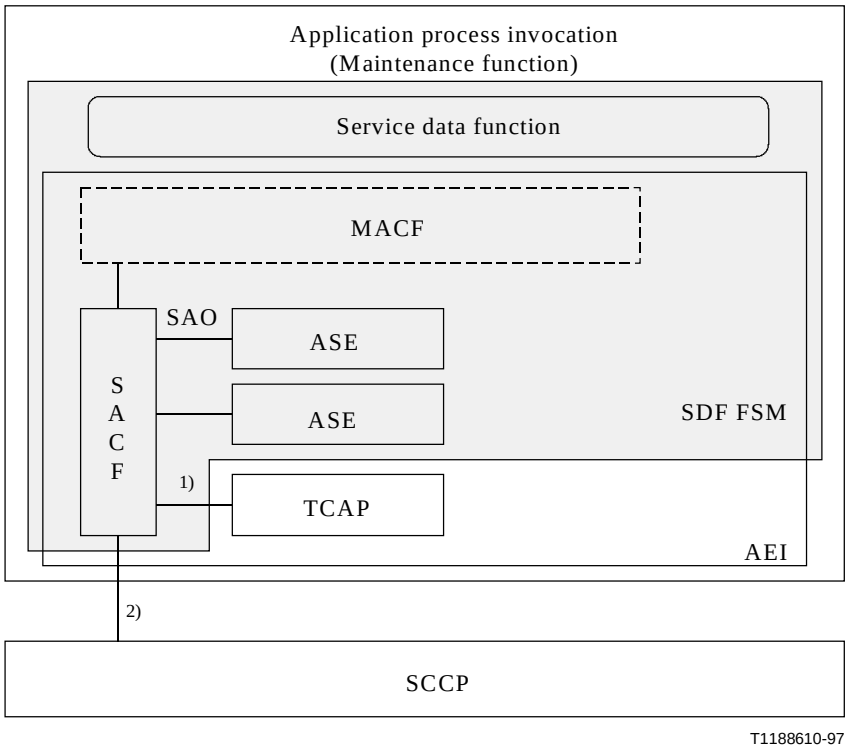
The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes TCAP (Transaction Capabilities Application Part) and one or more ASEs called TC-users, which are based on the Directory (X.500 series of Recommendations). The following subclauses define the TC-user ASE and SACF and MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed clause 17 shall be followed.

14.2 Model and interfaces

The functional model of the AE-SDF is shown in Figure 14-1; the ASEs interface to TCAP to communicate with the SCF and other SDFs, and interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 14-1.



AEI Application Entity Invocation
SDF Service Data Function
FSM Finite State Machine
SACF Single Association Control Function
SAO Single Association Object
MACF Multiple Association Control Function

1) TC-primitives.

2) N-primitives.

NOTE – The SDF FSM includes several finite state machines.

Figure 14-1/Q.1228 – Functional model of SDF

The interfaces shown in Figure 14-1 use the TC-user ASE primitives specified in Recommendation Q.771 [interface 1)] and N-Primitives specified in Recommendation Q.711 [interface 2)]. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clauses 4 to 10.

An instance of a specific SDF FSM may be created if IN call or non-call associated handling is received from the SCF, an SDF invokes or responds to a chaining operation to/from another SDF, or an SDF invokes or responds to a shadowing operation to/from another SDF.

The SDF FSM handles the interaction with the SCF FSM and another SDF FSM.

14.3 The SDF FSM structure

The structure of the SDF FSMs is illustrated in Figure 14-2.

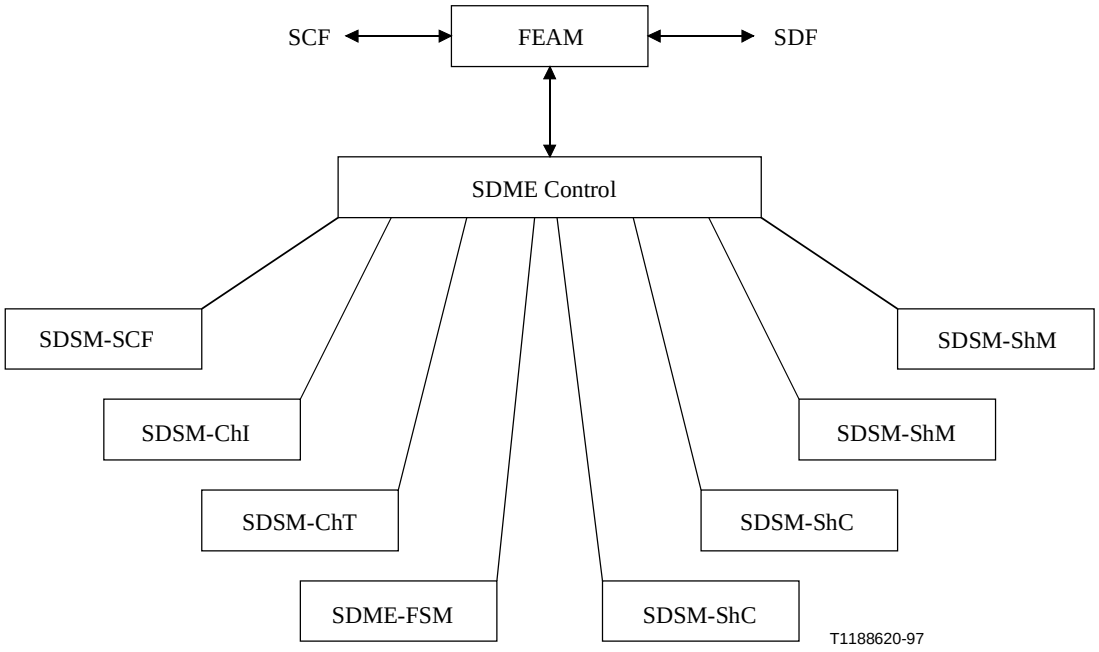


Figure 14-2/Q.1228 – SDF interfaces

The SDF FSM (SDSM-SCF) handles the interaction with the SCF FSM. The SDSM-ChI and SDSM-ChT handle the interactions between SDFs for chaining initiation and termination. The SDSM-ShSSi and SDSM-ShCSi handle the interactions between SDFs for shadowing supplier and consumer initiated by supplier. The SDSM-ShSCi and SDSM-ShCCi handle the same as above initiated by consumer. The SDME-FSM handles the interaction between the SDF and the SDF management functions.

The management functions related to the execution of operations received from the SCF or cooperating SDF are executed by the SDF Management Entity (SDME). The SDME is comprised of an SDME control and several instances of SDME-FSMs. The SDME control interfaces the different SDF FSMs (e.g. SDSM-SCF) and SDME-FSMs respectively as well as the Functional Entity Access Manager (FEAM).

The FEAM provides the low level interface maintenance functions including the following:

1) establishing and maintaining interfaces to the SCF and cooperating SDFs;

2) passing and queueing (when necessary) the messages received from the SCF and cooperating SDF to the SDME Control;

3) formatting, queueing (when necessary), and sending the messages received from the SDME Control to the SCF and cooperating SDF.

The SDME Control maintains the associations with the SCF and cooperating SDFs on behalf of all instances of the SDF FSMs (e.g. SDSM-SCF, SDSM-ChI). These instances of the SDF FSMs occur concurrently and asynchronously as SDF related events occur, which explains the need for a single entity that performs the task of creation, invocation and maintenance of the SDF FSMs. In particular, the SDME Control performs the following tasks:

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

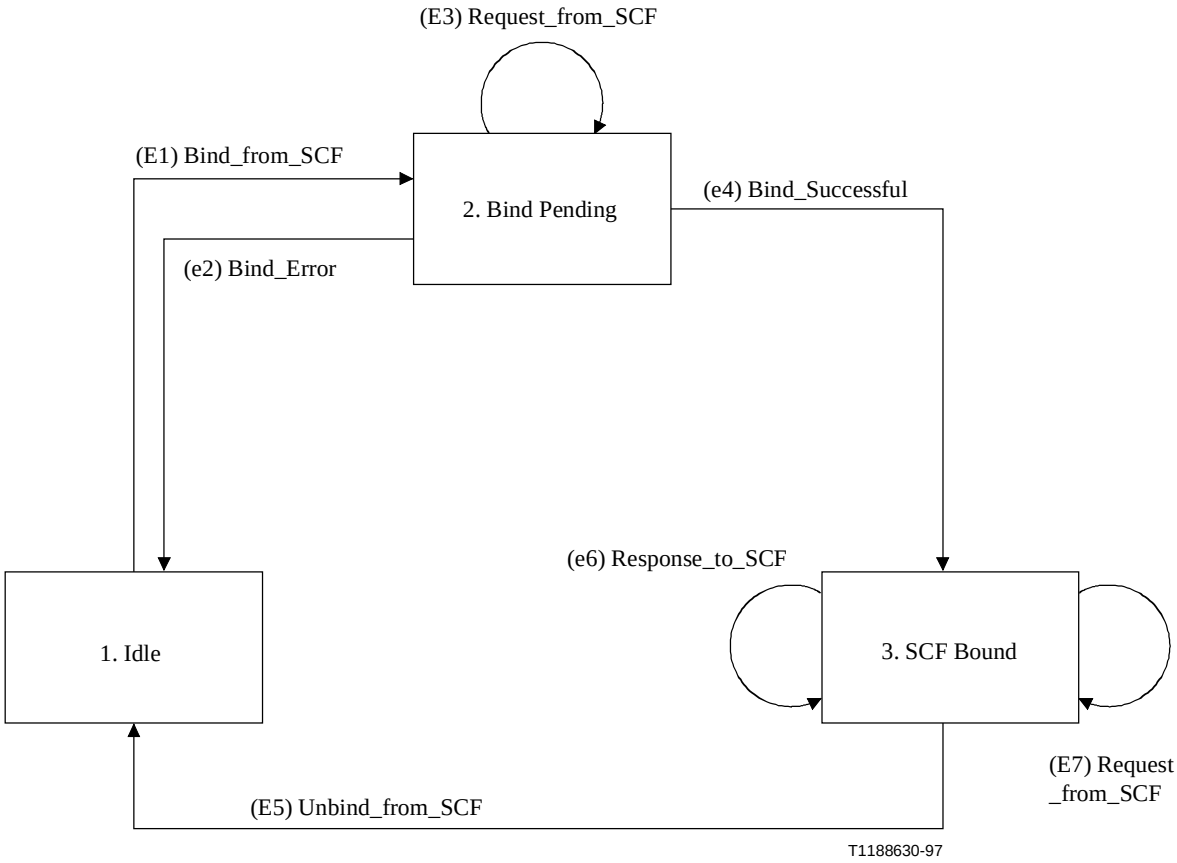
2) translates the SDF FSM outputs into corresponding messages to other FEs;

3) captures asynchronous activities related to management or supervisory functions in the SDF and creates an instance of an SDME-FSM. For example, management invocation of a shadowing procedure between network operators. In this case, the SDME Control will create an instance of the SDME-FSM to handle this management related operation.

14.4 SDF state transition models

14.4.1 SDF state transition model for SCF-related states

The SDF's job relating to interactions with the SCF is to (synchronously) respond to every request from the SCF after the Bind procedure. The respective Finite State Model (FSM) is depicted in Figure 14-3.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 14-3/Q.1228 – The SDF FSM

Each state is discussed in one of the following subclauses.

General rules applicable to more than one state are as follows:

In any state, if the dialogue with the SCF is terminated, then the SDF FSM returns to **Idle** state after ensuring that any resources allocated to the call have been de-allocated.

14.4.1.1 State 1: Idle

The only event accepted in this state is:

– (E1) Bind_from_SCF: This is an external event caused by the reception of directory Bind operation from the SCF. This event causes a transition out of this state to state 2, **Bind Pending**.

14.4.1.2 State 2: Bind Pending

In this state, a Bind request has been received from the SCF. The SDF is performing the SCF access control procedures behind the directoryBind operation (e.g. access authentication). There may also be a case such that the directoryBind operation is a dummy one. Then, the access authentication is not required. Three events are considered in this state:

– (e2) Bind_Error: This is an internal event, caused by the failure of the directoryBind operation previously issued to the SDF. This event causes a transition out of this state to state 1, **Idle** and a directoryBind error is returned to the invoking SCF;

– (E3) Request_from_SCF: This is an external event, caused by the reception of operations before the result from the directoryBind operation is determined.

  It involves one of the following operations:

  • search;
  • addEntry;
  • removeEntry;
  • modifyEntry;
  • execute.

  The operations are stored and the SDSM remains in the same state. When a transition occurs to another state, the operations are re-examined as if they had occurred in that state; and

– (e4) Bind_Successful: This is an internal event, caused by the successful completion of the directoryBind operation previously issued to the SDF. This event causes a transition out of this state to state 3, **SCF Bound**.

14.4.1.3 State 3: SCF Bound

In this state, the access of the SCF to the SDF was authorized and operations coming from the SCF are accepted. Besides waiting for requests from the SCF, the SDF can send in that state responses to previously issued operations. Three events are considered in this state:

– (E5) Unbind_from_SCF: This is an external event, caused by the reception of the in-directoryUnbind operation from the SCF. The SCF-SDF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**;

– (e6) Response_to_SCF: This is an internal event, caused either by the completion of the operations previously issued by the SCF or by generation of a referral error to the SCF. Responses or referrals are sent to the SCF. The SDSM remains in the same state; and

– (E7) Request_from_SCF: This is an external event, caused by the reception of a request from the SCF to the SDF.

  It involves one of the following operations:

  • search;
  • addEntry;
  • removeEntry;
  • modifyEntry;
  • execute.

  The SDSM remains in the same state.

14.4.2 SDF state transition model for SDF-related states

The SDF's job relating to interactions with other SDFs is to act upon shadowing and chaining operations. States related to SDF/SDF interactions for shadowing are given in 14.4.2.1. States related to SDF/SDF interactions for chaining are given in 14.4.2.2.

14.4.2.1 SDF state transition models for shadowing

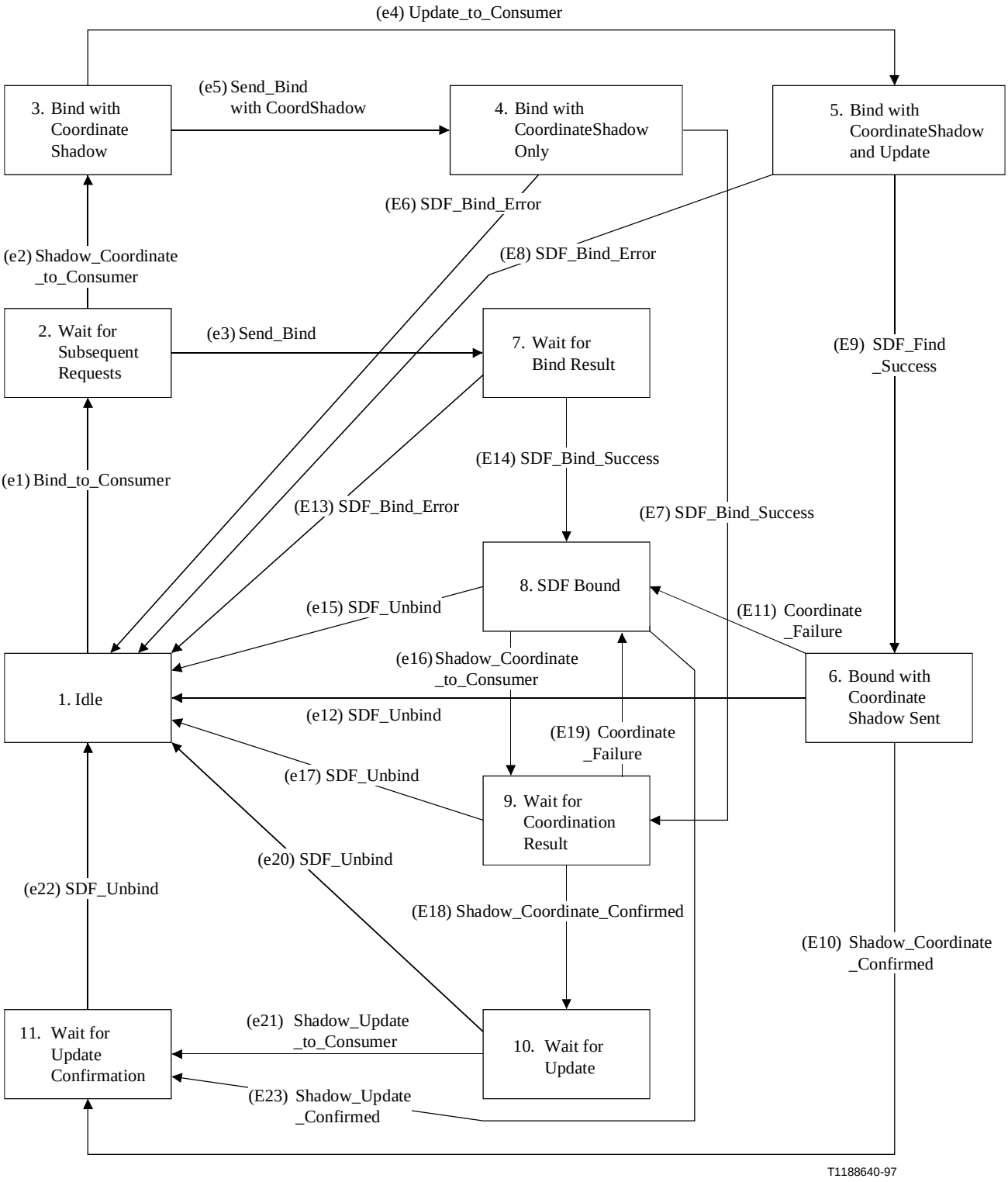
As for the shadowing procedure, an SDF can play the role of a copy supplier and a copy consumer. Moreover, the shadowing procedure can be initiated by a copy consumer as well as a copy supplier. Therefore, there can be in total four FSMs as described below.

The four different SDF FSMs could be gathered into one more complex FSM, but to clearly show the different roles played by an SDF, it was thought better to have four separate FSMs.

In the following FSMs, the possibility of sending the DSAShadowBind operation together with other DISP operations in one TC message is taken into consideration.

14.4.2.1.1 Shadow supplier-initiated supplier state machine (SDSM-ShSSi)

See Figure 14-4.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 14-4/Q.1228 – SDF FSM for a copy supplier in case of supplier-initiated (SDSM-ShSSi)

14.4.2.1.1.1 State 1: Idle

There is only one event accepted in this state:

– (e1) Bind_to_Consumer: This is an internal event caused by the request to execute a DSAShadowBind operation. This causes a transition out of this state to state 2, **Wait for Subsequent Requests**.

14.4.2.1.1.2 State 2: Wait for Subsequent Requests

In this state, a CoordinateShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the consumer is expected. The following two events are considered in this state:

– (e2) Shadow_Coordinate_to_Consumer: This is an internal event caused by the reception of a CoordinateShadowUpdate operation. The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition out of this state to state 3, **Bind with Coordinate Shadow**.

– (e3) Send_Bind: This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind operation is sent to the consumer SDF. This event causes a transition out of this state to state 7, **Wait for Bind Result**.

14.4.2.1.1.3 State 3: Bind with Coordinate Shadow

In this state, an UpdateShadow operation to be sent with the DSAShadowBind and CoordinateShadowUpdate operations, or a delimiter is expected. Two events are considered in this state:

– (e4) Update_to_Consumer: This is an internal event, caused by the reception of an UpdateShadow. This event causes a TCAP message containing the DSAShadowBind, CoordinateShadowUpdate and UpdateShadow operations to be sent to the consumer SDF. This event causes a transition out of this state to state 5, **Bind with CoordinateShadow and Update**.

– (e5) Send_Bind_with_CoordShadow: This is an internal event, caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind and CoordinateShadowUpdate operations is sent to the consumer SDF. This event causes a transition out of this state to state 4, **Bind with CoordinateShadow Only**.

14.4.2.1.1.4 State 4: Bind with CoordinateShadow Only

In this state, a DSAShadowBind result is expected from the consumer SDF. Two events are considered in this state:

– (E6) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to state 1, **Idle**.

– (E7) SDF_Bind_Success: This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to state 9,**Wait for Coordination Result**.

14.4.2.1.1.5 State 5: Bind with CoordinateShadow and Update

In this state, a DSAShadowBind result is expected from the consumer SDF. Two events are considered in this state:

– (E8) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to state 1, **Idle**.

– (E9) SDF_Bind_Success: This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to state 6, **Bound with Coordinate Shadow Sent**.

14.4.2.1.1.6 State 6: Bound with Coordinate Shadow Sent

In this state, a CoordinateShadowUpdate result is expected from the consumer SDF. Three events are considered in this state:

– (E10) Shadow_Coordinate_Confirmed: This is an external event, caused by the reception of a CoordinateShadowUpdate result. This indicates the successful completion of the CoordinateShadowUpdate operation previously issued to the consumer SDF. This event causes a transition out of this state to state 11, **Wait for Update Confirmation**.

– (E11) Coordinate_Failure: This is an external event, caused by the reception of an error to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to state 8, **SDF Bound**.

– (e12) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

14.4.2.1.1.7 State 7: Wait for Bind Result

In this state, a DSAShadowBind result is expected from the consumer. Two events are considered in this state:

– (E13) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to state 1, **Idle**.

– (E14) SDF_Bind_Success: This is an external event, caused by the successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to state 8, **SDF Bound**.

14.4.2.1.1.8 State 8: SDF Bound

In this state, the SDF has established an "authenticated association" to the consumer and is ready to send a CoordinateShadowUpdate operation to it. Two events are considered in this state:

– (e15) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to state 1, **Idle**.

– (e16) Shadow_Coordinate_to_Consumer: This is an internal event, that causes the sending of a request to a consumer SDF to coordinate the shadow to have it later updated. This event causes a transition out of this state to state 9, **Wait for Coordination Result**.

14.4.2.1.1.9 State 9: Wait for Coordination Result

In this state, the supplier SDF has sent a CoordinateShadowUpdate request and waits for the answer from the consumer SDF. Three events are considered in this state:

– (e17) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (E18) Shadow_Coordinate_Confirmed: This is an external event, caused by the reception of the response to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to state 10, **Wait for Update**.

– (E19) Coordinate_Failure: This is an external event caused by the reception of an error to the previously issued CoordinateShadowUpdate request operation. This event causes a transition back to state 8, **SDF Bound**.

14.4.2.1.1.10 State 10: Wait for Update

In this state, the supplier SDF has received a confirmation to the previously issued CoordinateShadowUpdate request and is ready to send an UpdateShadow request to the consumer SDF. Two events are considered in this state:

– (e20) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e21) Shadow_Update_to_Consumer: This is an internal event, that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to state 11, **Wait for Update Confirmation**.

14.4.2.1.1.11 State 11: Wait for Update Confirmation

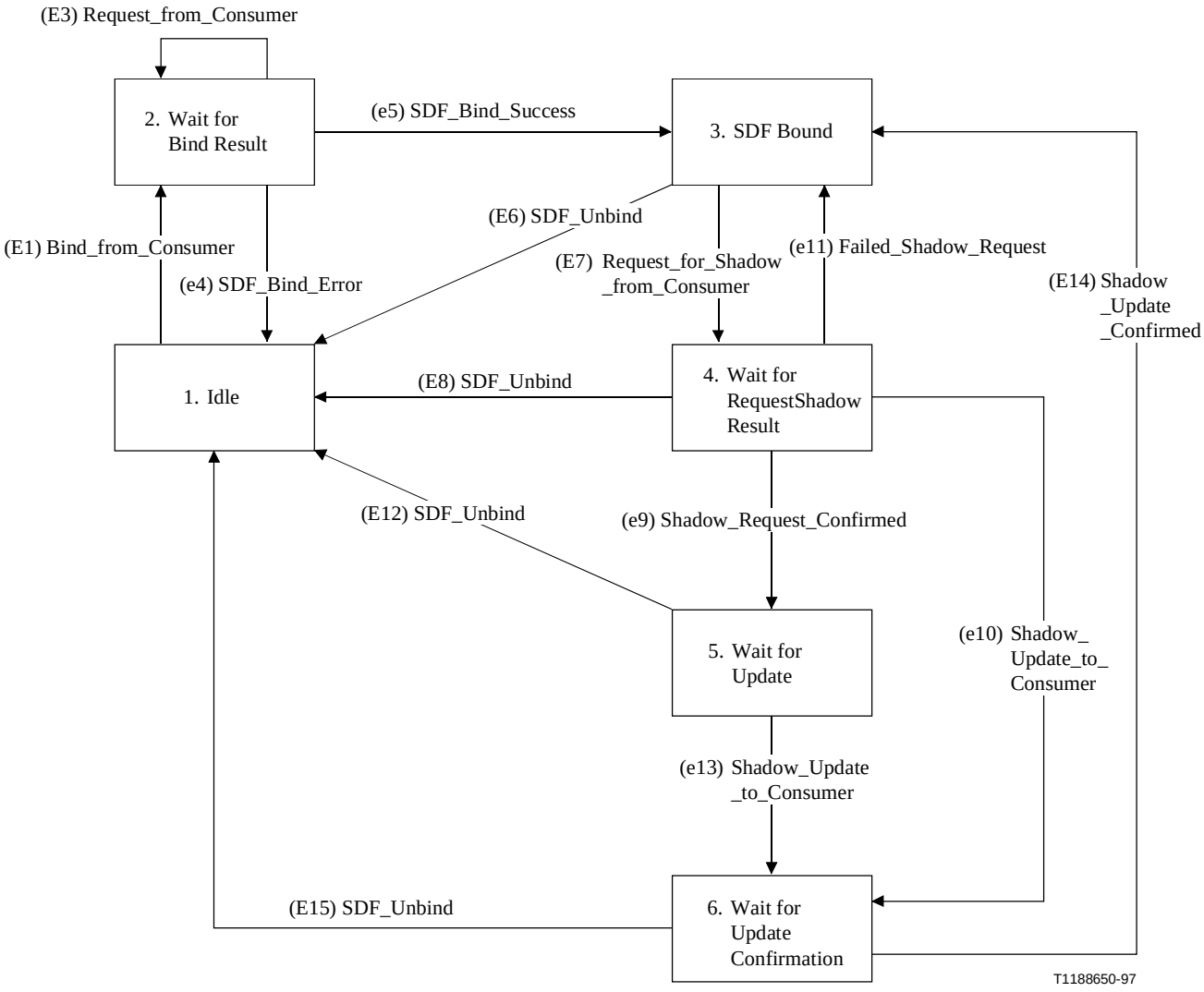
In this state, the supplier SDF has sent a UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

– (e22) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (E23) Shadow_Update_Confirmed: This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to state 8, **SDF Bound**.

14.4.2.1.2 Shadow consumer-initiated supplier state machine (SDSM-ShSCi)

See Figure 14-5.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 14-5/Q.1228 – SDF FSM for a copy supplier in case of consumer-initiated (SDSM-ShSCi)

14.4.2.1.2.1 State 1: Idle

One event is accepted in this state:

– (E1) Bind_from_Consumer: This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to state 2, **Wait for Bind Result**.

14.4.2.1.2.2 State 2: Wait for Bind Result

In this state, a DSAShadowBind operation is being performed. Three events are considered in this state:

– (E3) Request_from_Consumer: This is an external event, caused by the reception of a RequestShadowUpdate operation before the result from the DSAShadowBind operation is determined. This occurs when the RequestShadowUpdate is sent in the same TCAP message as the bind request. The RequestShadowUpdate message is stored and a transition occurs to

the same state. When a transition occurs to the next state, the RequestShadowUpdate is re-examined as if it occurred in that state.

– (e4) SDF_Bind_Error: This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the consumer. A DSAShadowBind error is returned. This event causes a transition out of this state to state 1, **Idle**.

– (e5) SDF_Bind_Success: This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the consumer. This event causes a DSAShadowBind result to be returned, and a transition out of this state to state 3, **SDF Bound**.

14.4.2.1.2.3 State 3: SDF Bound

In this state, the supplier SDF is expecting a RequestShadowUpdate operation from the consumer SDF. Two events are considered in this state:

– (E6) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to state 1, **Idle**.

– (E7) Request_for_Shadow_from_Consumer: This is an external event, caused by the reception of a RequestShadowUpdate operation from the consumer SDF. This event causes a transition out of this state to state 4, **Wait for RequestShadow Result**.

14.4.2.1.2.4 State 4: Wait for RequestShadow Result

In this state, the supplier SDF has received a RequestShadowUpdate operation from the consumer SDF. Four events are considered in this state:

– (E8) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e9) Shadow_Request_Confirmed: This is an internal event, that signals that the update agreement is acceptable. This causes the sending of a response to the previously received RequestShadowUpdate operation by the supplier SDF. This event causes a transition out of this state to state 5, **Wait for Update**.

– (e10) Shadow_Update_to_Consumer: This is an internal event, that signals that the update agreement is acceptable, and an UpdateShadow message is ready to be sent. This causes the sending of both a RequestShadowUpdate result and an UpdateShadow operation in the same TCAP message. This event causes a transition out of this state to state 6, **Wait for Update Confirmation**.

– (e11) Failed_Shadow_Request: This is an internal event, caused by the sending of an error to a previously received RequestShadowUpdate operation. This event causes a transition out of this state to state 3, **SDF Bound**.

14.4.2.1.2.5 State 5: Wait for Update

In this state, the supplier SDF has sent a response to the previously received RequestShadowUpdate operation and is ready to send an UpdateShadow operation to the consumer SDF. Two events are considered in this state:

– (E12) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e13) Shadow_Update_to_Consumer: This is an internal event that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to state 6, **Wait for Update Confirmation**.

14.4.2.1.2.6 State 6: Wait for Update Confirmation

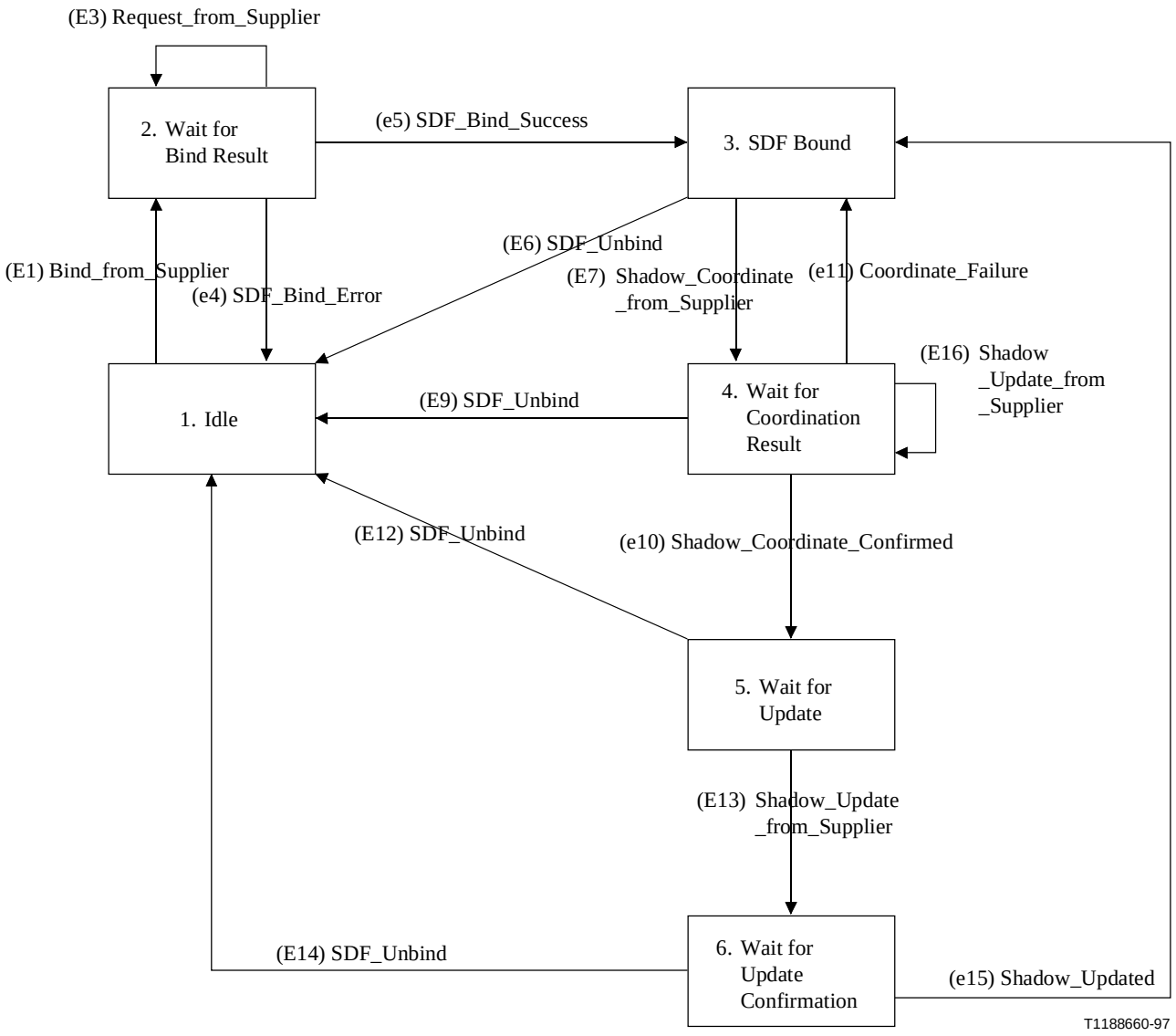
In this state, the supplier SDF has sent an UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

– (E14) Shadow_Update_Confirmed: This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to state 3, **SDF Bound**.

– (E15) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle.**

14.4.2.1.3 Shadow supplier-initiated consumer state machine (SDSM-ShCSi)

See Figure 14-6.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 14-6/Q.1228 – SDF FSM for a copy consumer in case of supplier-initiated (SDSM-ShCSi)

14.4.2.1.3.1 State 1: Idle

One event is accepted in this state:

– (E1) Bind_from_Supplier: This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to state 2, **Wait for Bind Result**.

14.4.2.1.3.2 State 2: Wait for Bind Result

In this state, the consumer has received a DSAShadowBind operation and is answering that operation. Three events are considered in this state:

– (E3) Request_from_Supplier: This is an external event, caused by the reception of operations before the result from the DSAShadowBind operation is determined. This occurs when the CoordinateShadowUpdate or UpdateShadow are sent in the same TCAP message

as the bind request. These operations are stored and a transition occurs to the same state. When a transition occurs to the following states, these operations are re-examined as if they occurred in that state.

– (e4) SDF_Bind_Error: This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the supplier SDF. A DSAShadowBind error is returned. This event causes a transition out of this state to state 1, **Idle**.

– (e5) SDF_Bind_Success: This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the supplier SDF. This event causes a transition out of this state to state 3, **SDF Bound**.

14.4.2.1.3.3 State 3: SDF Bound

In this state, the consumer SDF is expecting a CoordinateShadowUpdate operation from the supplier SDF. Two events are considered in this state:

– (E6) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to state 1, **Idle**.

– (E7) Shadow_Coordinate_from_Supplier: This is an external event, caused by the reception of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to state 4, **Wait for Coordination Result**.

14.4.2.1.3.4 State 4: Wait for Coordination Result

In this state, the consumer SDF has received a CoordinateShadowUpdate operation from the supplier SDF and is processing it. Four events are considered in this state:

– (E9) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e10) Shadow_Coordinate_Confirmed: This is an internal event, caused by the successful completion of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to state 5, **Wait for Update**.

– (e11) Coordinate_Failure: This is an internal event, caused by the sending of an error to a previously received CoordinateShadowUpdate operation. This event causes a transition out of this state to state 3, **SDF Bound**.

– (E16) Shadow_Update_from_Supplier: This is an external event, caused by the reception of a UpdateShadow operation in the same TCAP message as a CoordinateShadowBind and CoordinateShadowUpdate. The UpdateShadow message is stored and a transition occurs to the same state. When a transition occurs to the next state, the UpdateShadow is re-examined as if it had occurred in that state.

14.4.2.1.3.5 State 5: Wait for Update

In this state, the SDF is expecting an UpdateShadow operation. Two events are considered in this state:

– (E12) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (E13) Shadow_Update_from_Supplier: This is an external event, caused by the reception of the UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to state 6, **Wait for Update Confirmation**.

14.4.2.1.3.6 State 6: Wait for Update Confirmation

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

– (E14) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e15) Shadow_Updated: This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a transition out of this state to state 3, **SDF Bound.**

14.4.2.1.4 Shadow consumer-initiated consumer state machine (SDSM-ShCCi)

See Figure 14-7.

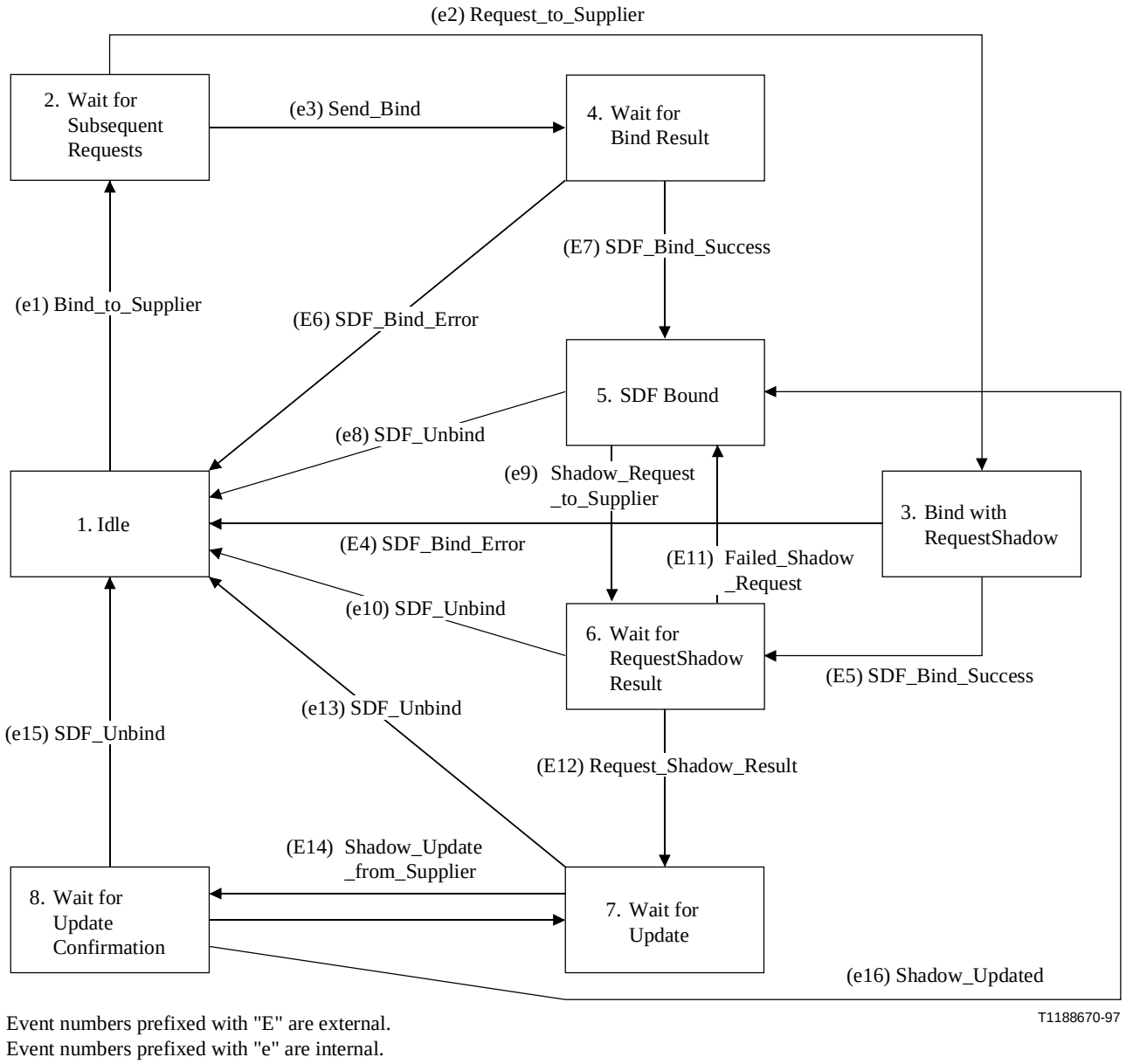


Figure 14-7/Q.1228 – SDF FSM for a copy consumer in case of consumer-initiated (SDSM-ShCCi)

14.4.2.1.4.1 State 1: Idle

There is only one event accepted in this state:

– (e1) Bind_to_Supplier: This is an internal event caused by the request to execute a DSAShadowBind operation. This event causes a transition out of this state to state 2, **Wait for Subsequent Requests**.

14.4.2.1.4.2 State 2: Wait for Subsequent Requests

In this state, a RequestShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the supplier is expected. The following two events are considered in this state:

– (e2) Request_to_Supplier: This is an internal event caused by the reception of a RequestShadowUpdate operation. This event causes a TCAP message containing the DSAShadowBind and RequestShadowUpdate operations to be sent to the supplier SDF. This event causes a transition out of this state to state 3, **Bind with RequestShadow**.

– (e3) Send_Bind: This is an internal event caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind operation is sent to the supplier SDF. This event causes a transition out of this state to state 4, **Wait for Bind Result**.

14.4.2.1.4.3 State 3: Bind with RequestShadow

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

– (E4) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to state 1, **Idle**.

– (E5) SDF_Bind_Success: This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to state 6, **Wait for RequestShadow Result**.

14.4.2.1.4.4 State 4: Wait for Bind Result

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

– (E6) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to state 1, **Idle**.

– (E7) SDF_Bind_Success: This is an external event, caused by the successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to state 5, **SDF Bound**.

14.4.2.1.4.5 State 5: SDF Bound

In this state, the consumer SDF is ready to send a RequestShadowUpdate operation to the supplier SDF. Two events are considered in this state:

– (e8) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to state 1, **Idle**.

– (e9) Shadow_Request_to_Supplier: This is an internal event, caused by the sending of a RequestShadowUpdate operation to the supplier SDF. This event causes a transition out of this state to state 6, **Wait for RequestShadow Result**.

14.4.2.1.4.6 State 6: Wait for RequestShadow Result

In this state, the consumer SDF has sent a RequestShadowUpdate operation and waits for the answer from the supplier SDF. Three events are considered in this state:

– (e10) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (E11) Failed_Shadow_Request: This is an external event, caused by the reception of an error to the previously issued RequestShadowUpdate operation. This event causes a transition out of this state to state 5, **SDF Bound**.

– (E12) Request_Shadow_Result: This is an external event, caused by the reception of the response to the previously issued RequestShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to state 7, **Wait for Update**.

14.4.2.1.4.7 State 7: Wait for Update

In this state, the consumer SDF has received a RequestShadowUpdate result and waits for an UpdateShadow operation from the supplier SDF. Two events are considered in this state:

– (e13) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1 **Idle**;

– (E14) Shadow_Update_from_Supplier: This is an external event caused by the reception of an UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to state 8 **Wait for Update Confirmation**.

14.4.2.1.4.8 State 8: Wait for Update Confirmation

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

– (e15) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to state 1, **Idle**.

– (e16) Shadow_Updated: This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a transition out of this state to state 5, **SDF Bound**.

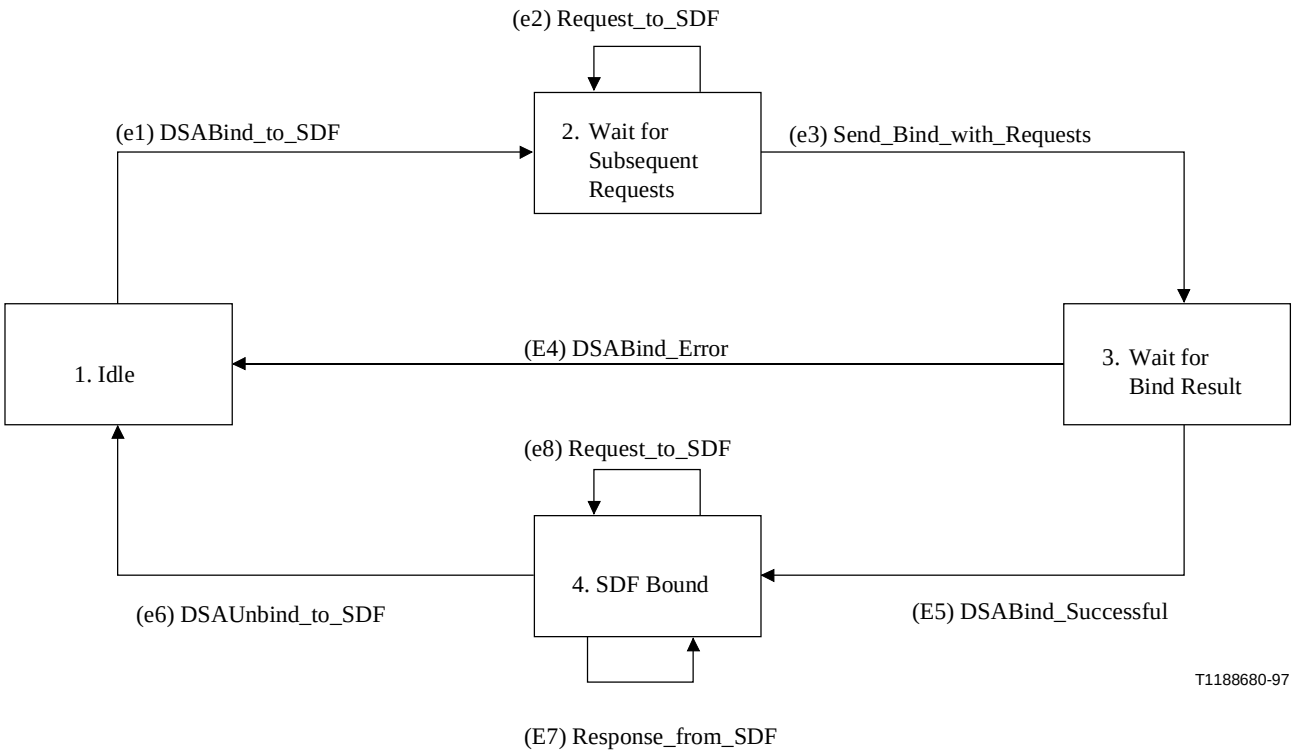
14.4.2.2 SDF state transition models for chaining

As for the chaining procedure, an SDF can act as a chaining initiator and as a chaining terminator. Therefore, there are two FSMs as described below.

In the following FSMs, the possibility of sending the DSABind operation together with other DSP operations in one TC message is taken into consideration.

14.4.2.2.1 SDF state transition models for chaining initiation (SDSM-ChI)

The Finite State Machine for an SDFs interaction with another SDF when acting as a chaining initiator is depicted in Figure 14-8.



Event numbers prefixed with "E" are external.
Event numbers prefixed with "e" are internal.

Figure 14-8/Q.1228 – SDF/SDF chaining initiator finite state machine (SDSM-ChI)

14.4.2.2.1.1 State 1: Idle

The only event accepted in this state is:

– (e1) DSABind_to_SDF: This is an internal event causing the sending of the DSABind operation to the SDSM-ChT. This event causes a transition out of this state to state 2, **Wait for Subsequent Requests**.

14.4.2.2.1.2 State 2: Wait for Subsequent Requests

In this state, subsequent operations to be sent with the DSABind operation (in the same message) to the SDSM-ChT are expected. The following two events are considered in this state:

– (e2) Request_to_SDF: This is an internal event causing the sending of an operation. It involves one of the following operations:

  • chainedSearch;
  • chainedAddEntry;
  • chainedRemoveEntry;
  • chainedModifyEntry;
  • chainedExecute.

  The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition to the same state;

– (e3) Send_Bind_with_Requests: This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing those arguments of the DSABind operation and other

operations, if any, are sent to the SDSM-ChT. This event causes a transition out of this state to state 3, **Wait for Bind Results**.

14.4.2.2.1.3 State 3: Wait for Bind Result

In this state, a DSABind request has been sent to the SDSM-ChT. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g. access authentication). Two events are considered in this state:

– (E4) DSABind_Error: This is an external event caused by the failure of the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to state 1, **Idle**.

– (E5) DSABind_Successful: This is an external event caused by the reception of the DSABind confirmation for the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to state 4, **SDF Bound**.

14.4.2.2.1.4 State 4: SDF Bound

In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized, chained operations can be sent to the SDSM-ChT, and results of chained operations coming from the SDSM-ChT are accepted. Three events are considered in this state:

– (e6) DSAUnbind_to_SDF: This is an internal event, causing the sending of the in-DSAUnbind operation to the SDSM-ChT. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**.

– (E7) Response_from_SDF: This is an external event, caused either by the reception results of the operations previously issued by the SDSM-ChI or reception of a referral from the SDSM-ChT. The SDSM-ChI remains in the same state.

– (e8) Request_to_SDF: This is an internal event, causing the sending of a chained operation to the SDSM-ChT.

  It involves one of the following operations:

  - chainedSearch;
  - chainedAddEntry;
  - chainedRemoveEntry;
  - chainedModifyEntry;
  - chainedExecute.

  The SDSM-ChI remains in the same state.

14.4.2.2.2 SDF state transition models for chaining termination (SDSM-ChT)

The Finite State Machine for an SDF interaction with another SDF when acting as a chaining terminator is depicted in Figure 14-9.

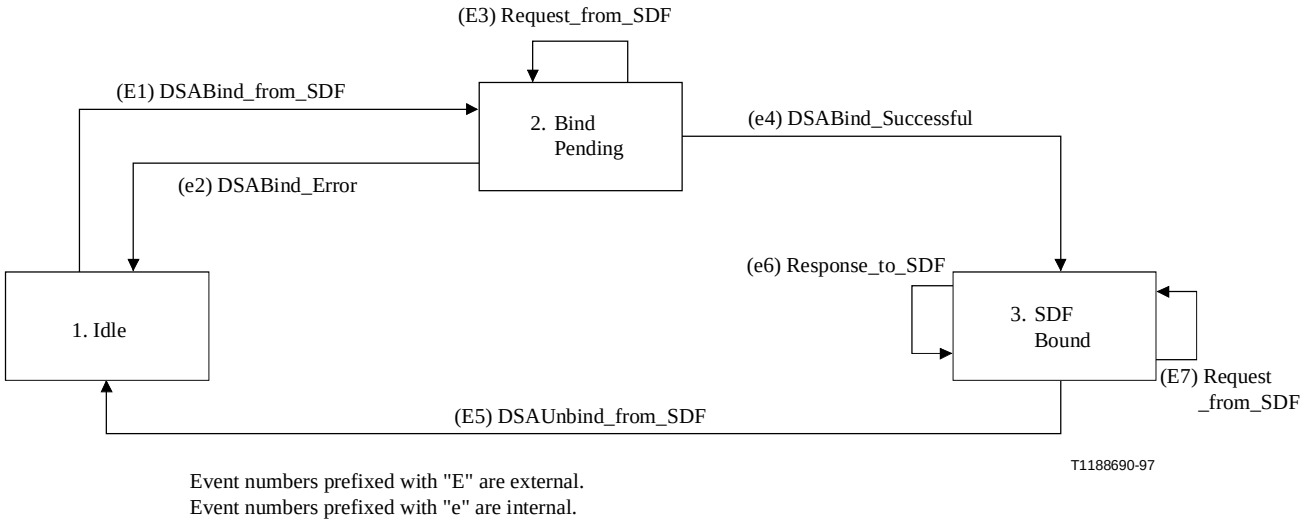


Figure 14-9/Q.1228 – SDF/SDF chaining terminator finite state machine (SDSM-ChT)

14.4.2.2.2.1 State 1: Idle

The only event accepted in this state is:

– (E1) DSABind_from_SDF: This is an external event caused by the reception of the DSABind operation from an SDSM-ChI. This event causes a transition out of this state to state 2, **Bind Pending**.

14.4.2.2.2.2 State 2: Bind Pending

In this state, a DSABind request has been received from the SDSM-ChI. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g. access authentication). Three events are considered in this state:

– (e2) DSABind_Error: This is an internal event caused by the failure of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to state 1 **Idle**, and a Bind error is returned to the SDSM-ChI;

– (E3) Request_from_SDF: This is an external event, caused by the reception of operations before the result from the DSABind operation is determined.

It involves one of the following operations:

- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry;
- chainedExecute.

The operations are stored and the SDSM-ChT remains in the same state. When a transition occurs to another state, the operations are re-examined as if they had occurred in that state; and

– (e4) DSABind_Successful: This is an internal event caused by the successful completion of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to state 3, **SDF Bound**.

14.4.2.2.2.3 State 3: SDF Bound

In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized and chained operations coming from the SDSM-ChI are accepted. Besides waiting for requests from the SDSM-ChI, the SDSM-ChT can send in this state responses to previously issued operations. Three events are considered in this state:

– (E5) DSAUnbind_from_SDF: This is an external event, caused by the reception of the in-DSAUnbind operation from the SDSM-ChI. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to state 1, **Idle**.

– (e6) Response_to_SDF: This is an internal event, caused either by the completion of operations previously issued by the SDSM-ChI or generation of a referral to the SDSM-ChI. Responses/referrals are sent to the SDSM-ChI. The SDSM-ChT remains in the same state.

– (E7) Request_from_SDF: This is an external event, caused by the reception of a request from the SDSM-ChI.

  It involves one of the following operations:

  • chainedSearch;
  • chainedAddEntry;
  • chainedRemoveEntry;
  • chainedModifyEntry;
  • chainedExecute.

  The SDSM-ChT remains in the same state.

15 CUSF application entity procedures

15.1 General

This clause provides the definition of the CUSF application entity (AE) procedures related to the CUSF-SCF interface. The procedures are based on the use of Common Channel Signalling System No. 7 (CCSS No. 7); other signalling systems can be used.

Capabilities not explicitly covered by these procedures may be supported in an implementation dependent manner in the SSP, CUSP, SN, while remaining in line with clause 2.

The AE, following the architecture defined in the Recommendations Q.700, Q.771 and Q.1400, includes TCAP (Transaction Capabilities Application Part) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE which interfaces with TCAP using the primitives specified in Recommendation Q.771; other signalling systems may be used.

The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

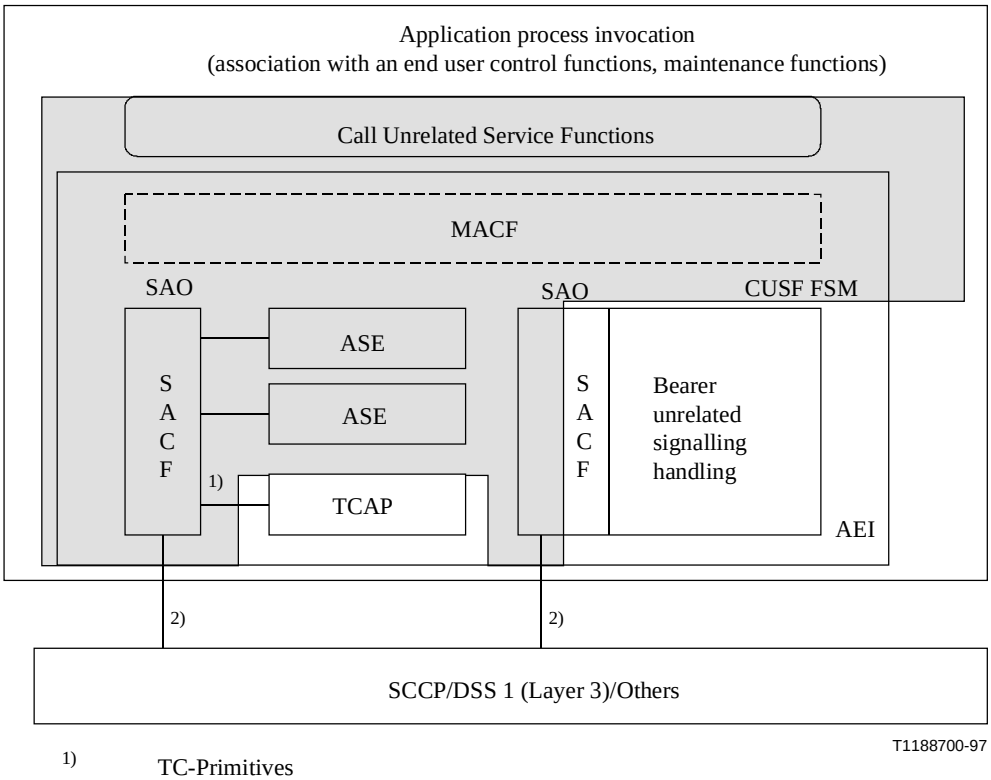
In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed clauses 17 and 18 shall be followed.

15.2 Model and interfaces

The functional model of the AE-CUSF is shown in Figure 15-1; the ASEs interface to TCAP to communicate with the SCF, and interface to the Call Control Function (CCF), the Service Switching Function (SSF), and the maintenance functions already defined for switching systems if needed. The

scope of this Recommendation is limited to the shaded area in Figure 15-1.

The interfaces shown in Figure 15-1 use the TC-user ASE primitives specified in Recommendation Q.771 and N-Primitives specified in Recommendation Q.711. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clauses 3 to 10.



1) TC-Primitives

2) N-Primitives

AEI Application Entity Invocations
CUSF Call Unrelated Service Function
FSM Finite State Machine
MACF Multiple Association Control Function
SACF Single Association Control Function
SAO Single Association Object

Figure 15-1/Q.1228 – Functional model of CUSF AE

15.2.1 Background for the modelling and protocol

The BCUSM depicts the handling of association and supplementary service processing for the Call unrelated case.

The actual processing can be modelled with various approaches. For example, the BCSM models only the basic call processing within the switch. This is because it is not possible to model every supplementary service; however, the common procedures like the user/terminal authentication can be modelled within the BCSM (Recommendation Q.1204).

Excepting for the association handling part, the interaction provides the ROSE APDU transmission on a UNI to support the supplementary services. The ROSE base interaction (e.g. DSS 1 ROSE) employs the common concept for the transfer syntax. Operation code, invoke ID, ROSE APDU, etc. are embedded in it to properly handle each invocation. Other parts like the parameters within the each APDU or the actions specified with the APDU (~ operation) take various specifications, so it is difficult to model all possibilities. In addition, the more detailed model requires the more frequent update of the model to introduce new procedures, and not a desirable situation for the IN.

15.2.2 Modelling and protocol

Although it is possible to model the analysis of the received ROSE APDU and invoke ID etc. and the invocation reject based on the analysis of them, the main aspect is how to model the handling of the ROSE APDUs, because, as mentioned in the previous subclause, they are service specific and cannot be explicitly modelled except for the general parts (component handling and association handling).

The state model for the interaction should depict the details of the activities in the physical entity to an extent which could be determined as for the BCSM, but the interaction is different from the circuit mode switched bearer services (represented by the BCSM) with the following:

- the variation of the service triggering points (corresponds to TDP) is limited to the association establishment/release phases or the reception of the ROSE APDU;
- the ROSE APDU is received during the association establishment/release phase or within the established association;
- the modelling of the analysis of a received ROSE APDU may be not necessary, because it can be well modelled as the TDP criteria check;
- the procedures with the interaction vary from service to service and have many variations, so each supplementary service has different states to handle the ROSE APDUs (but, reject or failure of the invocation may be possible to be modelled).

Considering these points and the background, the current BCUSM only models the association handling parts and the detection of the ROSE APDU reception. The PIA (Point in Association) indicates the association handling status. The DP indicates the place of:

i) the event of the association establishment request or release;

ii) the detection of the ROSE APDU reception. (Figure 15-2 shows what part is service-dependent and what part is general for this type of interaction.)

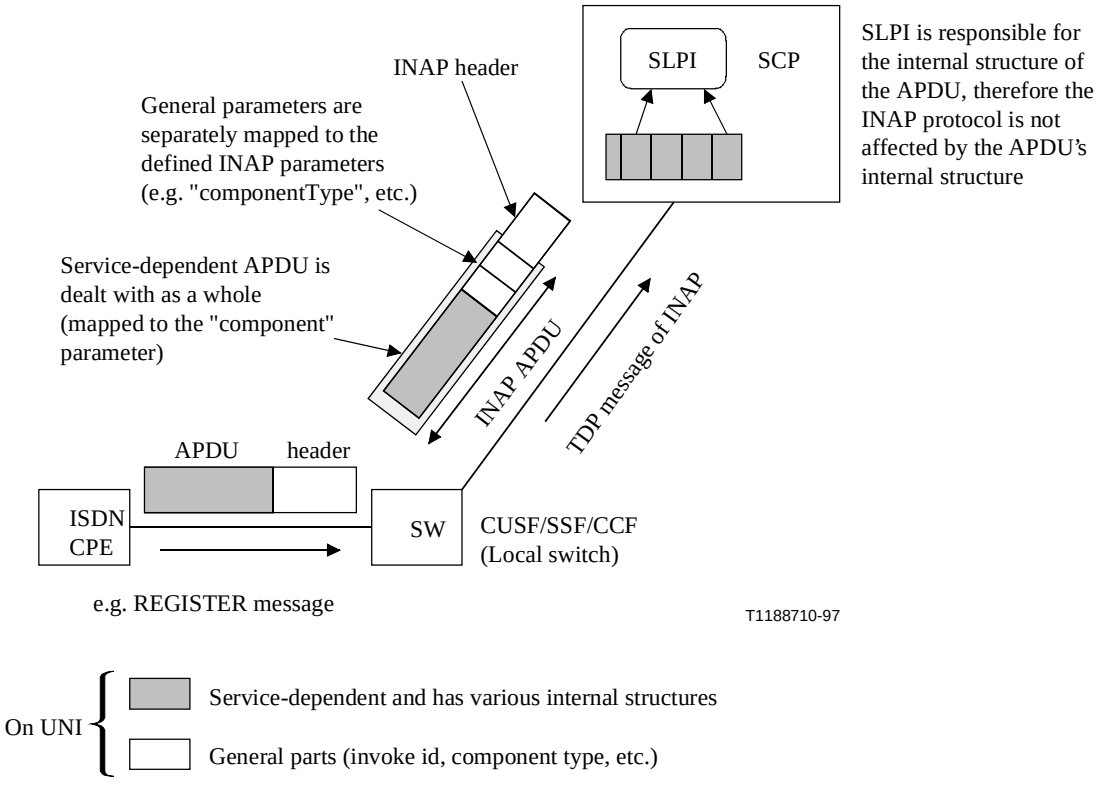


Figure 15-2/Q.1228 – ROSE APDU treatment scheme

15.3 Relations between CUSF FSM and the SSF/CCF and maintenance functions

The primitive interface between the CUSF FSM and the CCF/SSF/maintenance functions is an internal interface and is not subject to standardization in IN CS-2. Nevertheless, this interface should be in line with the BCUSM defined in clause 8/Q.1224.

The relationship between the BCUSM and the CUSF FSM may be described as follows for the case of a call unrelated associated association/operation attempt from an end user or IN service logic, or initiated by an end user or IN service logic:

– When a call unrelated associated association / operation attempt is initiated by an end user and processed at an exchange, an instance of a BCUSM is created. As the BCUSM proceeds, it encounters Detection Points (DPs, see clause 8/Q.1224). If a DP is armed as a Trigger DP (TDP), an instance of an CUSF FSM is created.

– If an InitiateAssociation is received from the SCF, an instance of a BCUSM is created, as well as an instance of a CUSF FSM.

The CUSF logic should:

– perform the DP processing actions specified in clause 8/Q.1224, including if DP criteria are met;

– check for SCF accessibility;

– handle service feature interactions in conjunction with the SSF.

The CUSF hands control back to the SSF at least in the following cases:

– if a trigger (TDP) criteria match is not found (e.g. insufficient information to proceed): the CUSF logic returns supplementary service control to the SSF;

– if the association is abandoned: the CUSF logic returns supplementary service control to the SSF and continues processing as described in 15.5;

– if the destination SCF is not accessible: the CUSF logic will apply final treatment to the end user for the TDP-R case, or return supplementary service control to the SSF for the TDP-N case.

The management functions related to the execution of operations received from the SCF are executed by the CUSF Management Entity (CUSME). The CUSME comprises a CUSME-Control and several instances of CUSME FSMs. The CUSME-control interfaces the different CUSF FSMs and CUSME FSMs respectively and the Functional Entity Access Manager (FEAM). Figure 15-3 shows the CUSF Interfaces.

The Functional Entity Access Manager (FEAM) provides the low level interface maintenance functions including the following:

1) establishing and maintaining the interfaces to the SCF;

2) passing and queueing (when necessary) the messages received from the SCF to the CUSME-Control;

3) formatting, queueing (when necessary), and sending the messages received from the CUSME-Control to the SCF.

The CUSME-control maintains the dialogues with the SCF on behalf of all instances of the CUSF Finite State Model (FSM). These instances of the CUSF FSM occur concurrently and asynchronously as associations occur, which explains the need for a single entity that performs the task of creation, invocation, and maintenance of the CUSF FSMs. In particular the CUSME-control performs the following tasks:

1) interprets the input messages from other FEs and translates them into corresponding CUSF

FSM events;

2) translates the CUSF FSM outputs into corresponding messages to other FEs;

3) captures asynchronous (with processing association and/or operation request from the end user) activities related to management or supervisory functions in the CUSF and creates an instance of a CUSME FSM.

The CUSF FSM passes component handling instructions to the related instances of the BCUSM as needed. DPs may be dynamically armed as EDPs, requiring the CUSF FSM to remain active. At some point, further interaction with the SCF is not needed, and the CUSF FSM may be terminated while the BCUSM continues to handle the association as needed.

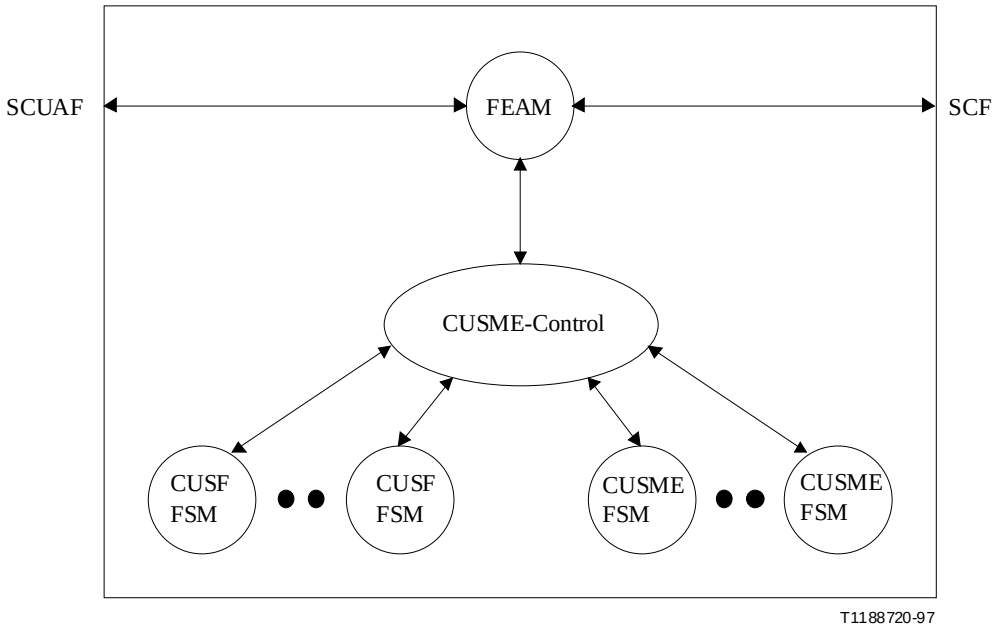


Figure 15-3/Q.1228 – CUSF Interfaces

15.4 CUSF management finite state model (CUSME FSM)

The CUSME FSM only relates to the ActivityTest, and for the operation the FSM only passes to the relevant CUSF FSM.

15.5 CUSF state transition diagram

Figure 15-4 shows the state diagram of the CUSF part of the SSP, CUSP, SN during the processing of an IN association request from the user or IN service logic/operation attempt.

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and is processed as follows:

– Process the operations in the order in which they are received.

– Each operation causes a state transition independent of whether or not a single operation or multiple operations are received in a message.

– The CUSF examines subsequent operations in the sequence. As long as sequential execution of these operations would leave the FSM in the same state, it will execute them. If a

subsequent operation causes a transition out of the state, then the following operations should be buffered until the current operation has been executed. In all other cases, await an event that would cause a transition out of the current state (such an event would be the completion of operation being executed, or reception of an external event). An example of this is as follows:

The CUSF receives the operations SendComponent, ReleaseAssociation in a component sequence inside a single TCAP message. Upon receipt of this message, these operations are executed up to and including SendComponent while the CUSF is in the Waiting For Instructions state. However, if the SendComponent specifies to deliver the component with an association release message to the user, the actual transmission of the component should be deferred until the next operation, ReleaseAssociation, will be executed when the FSM makes transition to the Idle state.

– If there is an error in processing one of the operations in the sequence, the CUSF FSM processes the error (see below) and discards all remaining operations in the sequence.

– If an operation is not understood or is out of context (i.e. violates the SACF rules defined by the CUSF FSM) as described above, ABORT the interaction.

In any state, if there is an error in a received operation, the maintenance functions are informed and the CUSF FSM remains in the same state as when it received the erroneous operation; depending on the class of the operation, the error could be reported by the CUSF to the SCF using the appropriate component (Recommendation Q.774).

In any state (except Idle), if the association requesting party abandons the association before it is established (i.e. before the Active PIA in the BCUSM), then the CUSF FSM should clear the association and ensure that any CUSF and CCF resources allocated to the association have been de-allocated, then moves to the Idle state. In any state (except Idle), if a call party releases a stable association (i.e. from the Active PIA in the BCUSM), then the CUSF FSM should move to the Idle state.

The CUSF has an application timer, T_{CUSF}, whose purpose is to prevent excessive association processing suspension time and to guard the association between the CUSF and the SCF.

Timer T_{CUSF} is set in the following cases:

– when the CUSF sends DP specific (see 15.5.2, State b: Waiting For Instructions).

On expiration of T_{CUSF} the CUSF FSM transits to the Idle state, and aborts the interaction with the SCF, and the CUSF progresses the BCUSM if possible.

The CUSF state diagram contains the following transitions (events):

er1 TDP-R encountered.

er2 Idle return from waiting for instructions.

er3 Request to send a component received (if no EDP armed) or monitoring instruction received.

er4 TDP-N encountered.

er5 Request to send a component received [if EDP(s) armed].

er6 EDP-N not last encountered.

er7 EDP-N last encountered.

er8 EDP-R encountered.

er10 Initiate association received.

The CUSF state diagram contains the following states:

– State a: Idle.

– State b: Waiting For Instructions.

– State c: Monitoring.

See Figure 15-4.

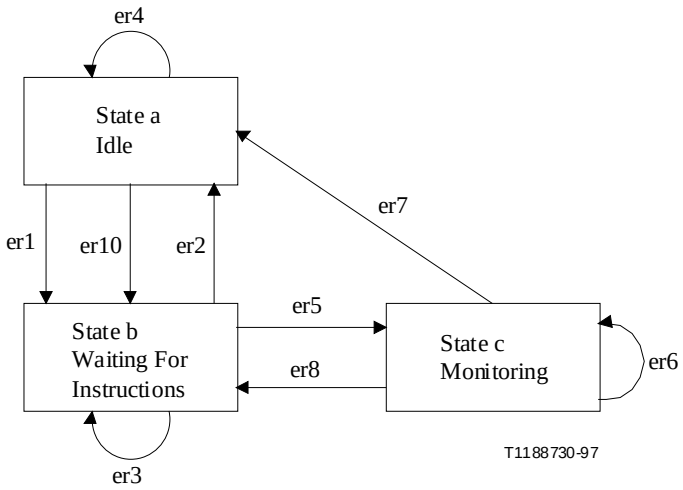


Figure 15-4/Q.1228 – CUSF FSM

15.5.1 State a: Idle

The CUSF FSM enters the Idle state when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state.

The CUSF FSM enters the Idle state when one of the following occurs:

– when the association is released by the end user request in the Waiting For Instructions (transition er2) or in the Monitoring (transition er7);

– when a ReleaseAssociation operation is processed in the Waiting For Instructions (transition er2);

– when a last EDP-N is reported in the Monitoring (transition er7);

– when the application timer T_{CUSF} expires in the Waiting for Instructions state (transition er2).

When transiting to the Idle state, if there is a component to be delivered with an association release message to the user, the CUSF sends the component with the specified association release message to the SCUAF before returning to Idle.

During this state, the following call unrelated associated event can occur:

– an armed TDP is encountered related to a possible IN call unrelated attempt, the CUSF FSM acts as described below:

  • if the DP is a TDP-N, send a DP-specific operation to the SCF, as determined from DP processing; there is no resulting transition to a different state (transition er4);

  • if the DP is a TDP-R, send a DP-specific operation to the SCF, as determined from DP processing, and transit to the Waiting For Instructions state (transition er1);

– a message related to a new transaction containing an InitiateAssociation operation is received from the SCF: in this case the CUSF moves to the state Waiting For Instructions (transition er10).

Any other operation received from the SCF while the CUSF is in Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TCAP (Recommendation Q.774).

NOTE – *DP-specific* operations are the following (see clause 10): ActivationReceivedAndAuthorized, ComponentReceived and AssociationReleaseRequested.

15.5.2 State b: Waiting For Instructions

This state is entered from the Idle state, as indicated above (transition er1), .

In this state the CUSF FSM is waiting for an instruction from the SCF; association handling /supplementary service processing is suspended and an application timer (T_{CUSF}) should be set on entering this state.

During this state, the following events can occur:

– The user releases the association. This should be processed in accordance with the general rules in 15.5.

– The application Timer T_{CUSF} expires: the CUSF FSM moves to the Idle state, the SSF processes the invocation if possible, the T_{CUSF} expiration is reported to the maintenance functions and the transaction is aborted.

– An operation is received from the SCF: The CUSF FSM acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the CUSF with no resulting transition to a different state (transition er3):

SendComponent (if no EDP is armed);

RequestReportBCUSMEvent.

The following operation may be received from the SCF and processed by the CUSF, causing a state transition to the Monitoring state (transition er5):

SendComponent [if EDP(s) is(are) armed]

ReleaseAssociation operation may be received from the SCF. In this case, the CUSF FSM should release the association to the user and ensure that any CUSF resources allocated to the association have been de-allocated (transition er2).

Any other operation received in this state should be processed in accordance with the general rules in 15.5.

15.5.3 State c: Monitoring

The CUSF enters this state from the Waiting For Instructions state (transition er5) upon receiving a SendComponent if EDP(s) is (are) armed.

In this state, the timer T_{CUSF} is not used; i.e., the expiration of T_{CUSF} does not have any impact on the CUSF FSM.

During this state, the following events can occur:

– An EDP-N should be reported to the SCF by sending a DP-specific operation; the CUSF FSM should remain in the Monitoring state (transition er6) if one or more EDPs are armed. The CUSF FSM should move to the Idle state (transition er7) if there are no remaining EDPs armed.

– An EDP-R should be reported to the SCF by sending a DP-specific operation; the CUSF FSM should move to the Waiting For Instructions state (transition er8).

– The receipt of an END or ABORT primitive from TCAP has no effect on the association; the association may continue or be completed with the information available. In this case, the CUSF FSM transits to the Idle state (transition er7), disassociating the CUSF FSM from the association.

– The user abandons or releases association. This should be processed in accordance with the general rules in 15.5.

16 Error procedures

This clause defines the generic error procedures for the IN CS-1 INAP. The error procedure descriptions have been divided in two subclauses, subclause 16.1 listing the errors related to INAP operations and subclause 16.2 listing the errors related to error conditions in the different FEs which are not directly related to the INAP operations.

16.1 Operation related error procedures

The following subclauses define the generic error handling for the operation related errors. The errors are defined as operation errors in clauses 4 to 10. The TCAP services which are used for reporting operation errors are described in 18.1.

Errors which have a specific procedure for an operation are described in clauses 11 to 15 with the detailed procedure of the related operation.

All errors, which can be detected by the ASN.1 decoder, already may be detected during the decoding of the TCAP message and indicated by the TC error indication "MistypedParameter" in the TC-U-Reject.

16.1.1 AttributeError

16.1.1.1 General description

16.1.1.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report an attribute related problem. The conditions under which an attribute error is to be issued are defined in Recommendation X.511 (1993) subclause 12.4.

16.1.1.1.2 Argument description

The attribute error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.4.

16.1.1.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
Search

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.

Postcondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the SCF. After returning the error, no further error treatment is performed.

16.1.1.3 Operations SDF→SDF

ChainedAddEntry
ChainedExecute
ChainedModifyEntry
ChainedSearch

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

This error is reported to the SCF.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

b) *Returning Error*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the other SDF. After returning the error, no further error treatment is performed.

16.1.2 Cancelled

16.1.2.1 General description

16.1.2.1.1 Error description

The Error "Cancelled" gives an indication to the SCF that the cancellation, as it was requested by the SCF, of a specific Operation, has been successful. The SCF is only able to cancel certain predefined SCF→SRF Operations.

16.1.2.2 Operations SCF→SRF

PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage

Procedures at invoking entity (SCF)

a) *Sending Cancel*

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

The SCF sends a Cancel after a Play Announcement or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

b) *Receiving Cancelled Error*

Precondition: SCSM state any Service Logic dependent.

Postcondition: SCSM state any Service Logic dependent.

After sending a Cancel operation, the Service Logic may continue (e.g. sending more PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The Cancelled Error can therefore be received in any state. The treatment is Service Logic dependent.

Procedures at responding entity (SRF)

a) *Receiving Cancel*

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

The indicated PlayAnnouncement or PromptAndCollectUserInformation is terminated if it is presently executing or deleted from the buffer. If the indicated PlayAnnouncement or PromptAndCollectUserInformation is already executed, this causes a failure ("CancelFailed").

b) *Sending Cancel Error*

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

After returning the "Cancelled" Error the SRF stays in the same state. The execution of the indicated PlayAnnouncement or PromptAndCollectUserInformation is aborted, i.e. the SRF remains connected and the next PlayAnnouncement or PromptAndCollectUserInformation is executed if available.

16.1.3 CancelFailed

16.1.3.1 General description

16.1.3.1.1 Error description

This Error is returned by Cancel if the cancelling of an Operation, as requested by the SCF, was not successful. Possible failure reasons are:

0 unknownOperation, when the InvokeID of the operation to cancel is not known to SRF (this may also happen in case the operation has already been completed);

1 tooLate, when the invokeID is known but the execution of the operation is in a state that it cannot be Cancelled anymore. For instance, the announcement is finished but the SpecializedResourceReport has not been sent to the SCF yet. The conditions for the occurrence of failure reason "tooLate" may be implementation dependent;

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

16.1.3.1.2 Argument description

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

16.1.3.2 Operations SCF→SSF

Cancel
CancelStatusReportRequest

16.1.3.3 Operations SCF→SRF

Cancel

Procedures at invoking entity (SCF)

a) *Sending Cancel*

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

The SCF sends a Cancel after a Play Announcement or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

b) *Receiving CancelFailed Error*

Precondition: SCSM state any Service Logic dependent.

Postcondition: SCSM state any Service Logic dependent.

After sending a Cancel operation, the Service Logic may continue (e.g. sending another PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The CancelFailed Error can therefore be received in any state. The treatment is Service Logic dependent.

Procedures at responding entity (SRF)

a) *Receiving Cancel*

However, the indicated PlayAnnouncement or PromptAndCollectUserInformation is not known, or already executed. This causes a failure, CancelFailed.

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction; or
	SRSM state 1	Idle.

b) *Sending CancelFailed Error*

Precondition:	SRSM state 3	User Interaction; or
	SRSM state 1	Idle.
Postcondition:	SRSM state 3	User Interaction; or
	SRSM state 1	Idle.

After returning the CancelFailed, the SRF stays in the same state.

16.1.4 DSAReferral

16.1.4.1 General description

16.1.4.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to chaining to a chained operation. The conditions under which a DSAReferral error is to be issued are defined in Recommendation X.518 (1993) subclause 8.3.

16.1.4.1.2 Argument description

The DSAReferral error parameter and problem codes are specified in Recommendation X.518 (1993) subclause 13.2.

16.1.4.2 Operations SDF→SDF

ChainedAddEntry
ChainedExecute
ChainedModifyEntry
ChainedRemoveEntry
ChainedSearch

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

After receiving a DSAReferral error, the SDF can continue to execute the operation by accessing another SDF which is indicated by this error.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition: SDSM-ChT state 3 SDF Bound.

Postcondition: SDSM-ChT state 3 SDF Bound.

b) *Returning Error*

Precondition: SDSM-ChT state 3 SDF Bound.

Postcondition: SDSM-ChT state 3 SDF Bound.

The SDF could not perform the operation due to a chaining condition and therefore sends a DSAReferral error to another SDF. After returning the error, no further error treatment is performed.

16.1.5 ETCFailed

16.1.5.1 General description

16.1.5.1.1 Error description

ETCFailed is an error from SSF to SCF, indicating the fact that the establishment of a temporary connection to an assisting SSF or SRF was not successful (e.g. receiving a "Backwards Release" after sending an IAM).

16.1.5.2 Operations SCF→SSF

EstablishTemporaryConnection

Procedures at invoking entity (SCF)

a) *SCF sends EstablishTemporaryConnection to SSF*

Precondition: SCSM state 3.1 Determine Mode.

Postcondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

b) *SCF receives ETCFailed Error from SSF*

Precondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

Postcondition: SCSM state 2.1 Preparing SSF Instructions.

Error handling depends on the Service Logic, e.g. selecting another SRF or continuing the processing of the call.

Procedures at responding entity (SSF)

A SSF receives EstablishTemporaryConnection from a SCF but the establishment of the connection fails, results in returning an ETCFailed Error to the SCF.

Precondition: SSF FSM state c Waiting for Instructions.

Postcondition: SSF FSM state c Waiting for Instructions.

No further Error treatment.

16.1.6 ExecutionError

16.1.6.1 General description

16.1.6.1.1 Error description

Execution Error is an error sent from the SDF to SCF, indicating that the request to execute an entry method has failed. The failure can be due to an incorrect input value or the failure of an internal operation or data access logic associated with the execution of the entry method.

16.1.6.1.2 Argument description

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

16.1.6.2 Operations SCF→SDF

Execute

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 2	Bind Pending; or
	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

In the case that the SDF has not received the correct input value, the SDF sends an ExecutionError to the SCF with problem set to missingInputValues. In all other cases, the SDF attempts to execute the entry method but fails due to the failure of an internal operation or data access logic. The SDF therefore sends an ExecutionError to the SCF with problem set to executionFailure. After returning the error, the SDF is returned to the state it was prior to the operation, that is the execute operation is considered to be an atomic operation.

16.1.6.3 Operations SDF→SDF

ChainedExecute

Procedure at Invoking Entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

This error is reported to the SCF in case of chained operations.

Procedure at Responding Entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

b) *Returning Error*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

The SDF could not perform the operation and therefore sends an executionError to the other SDF. After returning the error, no further error treatment is performed.

16.1.7 ImproperCallerResponse

16.1.7.1 General description

16.1.7.1.1 Error description

The format of the user input has been checked by the SRF and does not correspond to the required format as it was defined in the initiating Operation.

16.1.7.2 Operations SCF→SRF

PromptAndCollectUserInformation
PromptAndReceiveMessage

Procedures at invoking entity (SCF)

a) *SCF sends PromptAndCollectUserInformation to SRF*

Precondition:	SCSM state 3.1	Determine Mode; PromptAndCollectUserInformation will accompany the ConnectToResource; or
	SCSM state 3.2	Waiting for AssistRequestInstructions; after EstablishTemporaryConnection; or
	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformations are active.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

b) *SCF receives ImproperCallerResponse Error from SRF*

Precondition: SCSM state 4.1 Waiting for Response from the SRF.
Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

Error treatment depends on Service Logic. A SCF can initiate new User Interaction or force a Disconnect (to SSF).

Procedures at responding entity (SRF)

a) *SRF receives PromptAndCollectUserInformation*

Precondition: SRSM state 2 Connected; or
SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

b) *response from caller is not correct, SRF returns ImproperCallerResponse to SCF*

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

SRF waits for a new Operation from SCF. This may be a new PromptAndCollectUserInformation or PlayAnnouncement.

16.1.7.3 Operations SCF→SCF

ChainedProvideUserInformation
ProvideUserInformation

Procedures at invoking entity (SCF)

a) *Sending operation*

Precondition: SCSM-Sup in Assisting Mode (in case of ProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation).

Postcondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation).

b) *Receiving Error*

Precondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation).

Postcondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation).

Once the supporting SCF or chaining terminator supporting SCF receives this type of error, it is waiting the reception of a SCFUnbind operation. A guard timer is armed.

Procedures at responding entity (SCF)

a) *Receiving operation*

Precondition: SCSM-Con in Assisted Mode (in case of ProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation).

Postcondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation).

b) *Returning Error*

Precondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation).

Postcondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation).

Once the controlling SCF issues this type of error, internal event (e12) SCFUnbind in SCSM-Con or event (e6) SCFUnbind in SCMS-ChI occurs.

16.1.8 MissingCustomerRecord

16.1.8.1 General description

16.1.8.1.1 Error description

The Service Logic Program could not be found in the SCF because the required customer record does not exist, or the requested Service Logic Program Instance, indicated by the correlationID in "AssistRequestInstructions", does not exist anymore. These two cases should be distinguished as two different error situations because the error procedure shows that the occurrence of the MissingCustomerRecord error is reported to the maintenance function, but the report to the maintenance function for the occurrence of the former case should be optional because it occurs not only in extraordinary situation but in ordinary situation. For example, the former may occur when the end user dials a missing freephone number.

16.1.8.2 Operations SSF→SCF

AnalysedInformation
AssistRequestInstructions
CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall

TNoAnswer
TSuspended

Procedures at invoking entity (SSF)

a) *Sending Operation*

Precondition:	SSF FSM state b	Trigger processing; or
	SSF FSM state b'	Waiting for Instructions; in case of assist/hand-off.
Postcondition:	SSF FSM state c	Waiting for Instructions; or
	SSF FSM state b'	Waiting for Instructions; in case of assist/hand-off.

b) *SSF receives Error "MissingCustomerRecord"*

Precondition:	SSF FSM state c	Waiting for Instructions; or
	SSF FSM state b'	Waiting for Instructions; in case of assist/hand-off.
Postcondition:	SSF FSM state a	Idle; or
	SSF FSM state a'	Idle; in case of assist/hand-off.

The CCF routes the call if necessary (e.g. default routing to a terminating announcement).

Procedures at responding entity (SCF)

Precondition:	1)	SCSM	appropriate state.
	2)	SCSM	Operation received, appropriate event occurred.
Postcondition:	1)	SCSM state 1	Idle; in case of all operations listed above, except AssistRequestInstructions.
	2)	SCSM state 2.1	Preparing SSF instructions; for AssistRequestInstructions.

The SCSM detects that the required Service Logic Program does not exist. The Service Logic Program Instance may not exist anymore (e.g. in case of the operation AssistRequestInstructions), or the Service Logic Program may have never existed at all (i.e. the customer record in the SCF does not exist, e.g. in case of TDPs a Service Logic Program is attempted to be invoked). The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed (however, it is optional for the TDP operation case).

16.1.8.3 Operations SRF→SCF

AssistRequestInstructions

Procedures at invoking entity (SRF)

a) *Sending Operation*

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 2	Connected.

b) *SRF receives Error "MissingCustomerRecord"*

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 1	Idle.

SRF initiated Disconnect.

Procedures at responding entity (SCF)

The SCSM detects that the required Service Logic Program does not exist (anymore). The establishment of a connection between the SSF and the SRF took too long or the correlationID was invalid. In both cases the requested Service Logic Program cannot be found. The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed.

16.1.8.4 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord);
or
SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting Mode (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in Assisting Mode (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

b) *Receiving Error*

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state Idle (in case of first HandlingInformationRequest); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Error treatments in controlling SCF or chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

a) *Receiving Operation*

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted Mode (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in Assisted Mode (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in Assisted Mode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

b) *Returning Error*

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Error treatments in controlling SCF or chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

16.1.8.5 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

a) *Sending Operation*

Precondition: CUSF FSM state a Idle.

Postcondition: CUSF FSM state b Waiting for Instructions.

b) *CUSF receives Error "MissingCustomerRecord"*

Precondition: CUSF FSM state b Waiting for Instructions.

Postcondition: CUSF FSM state a Idle.

The CUSF continues to handle the association or terminate the association with default procedures (network-operator specific).

Procedures at responding entity (SCF)

Precondition:	1)	FSM for CUSF	appropriate state (in State N1 Idle).
	2)	FSM for CUSF	Operation received, appropriate event occurred.
Postcondition:	1)	FSM for CUSF state N1	Idle.

The FSM for CUSF detects that the required Service Logic Program does not exist. This is the same situation as for the SSF-SCF case.

16.1.9 MissingParameter

16.1.9.1 General description

16.1.9.1.1 Error description

There is an Error in the received Operation argument. The responding entity cannot start to process the requested Operation because the argument is incorrect: a mandatory parameter (the application shall always return this error in case it is not detected by the ASN.1 decoder) or an expected optional parameter which is essential for the application is not included in the Operation argument.

16.1.9.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
Cancel
CancelStatusReportRequest
FurnishChargingInformation
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent
RequestReportFacilityEvent
RequestReportUTSI
ResetTimer
SendChargingInformation
SendFacilityInformation
SendSTUI

Call associated/call processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
ContinueWithArgument
CreateCallSegmentAssociation
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments

MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM	any state in which the above Call associated operations can be transferred.
	SCME	any state in which the above Non-call associated operations can transferred.
Postcondition:	SCSM	any state as result of the transfer of any of the above operations.
	SCME	any state as result of the transfer of any of the above Non-call associated operations.

b) *SCF receives Error "MissingParameter"*

Precondition:	SCSM	any state as result of the transfer of any of the above Call associated operations.
	SCME	any state as result of the transfer of any of the above Non-call associated operations.
Postcondition:	SCSM	transition to the initial state (i.e. before sending the erroneous operation).
	SCME	transition to the initial state (i.e. before sending the erroneous operation).

The Service Logic and maintenance functions are informed. Further treatment of the call is dependent on Service Logic.

Procedures at responding entity (SSF)

Precondition:	1)	SSF FSM	appropriate state.
	2)	SSF FSM	Call associated operation received, appropriate event occurred.
	3)	SSME	appropriate state.
	4)	SSME	Non-call associated operation received, appropriate event.
Postcondition:	1)	SSF FSM	transition to the same state.
	2)	SSME	transition to the initial state (i.e. before receiving the erroneous operation).

The SSF FSM detects the error in the received operation. The Error parameter is returned to inform the SCF of this situation.

16.1.9.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
AssistRequestInstructions
CollectedInformation
FacilitySelectedAndAvailable

InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer
TSuspended

Procedures at invoking entity (SSF)

a) *Sending Operation*

Precondition:	SSF FSM	any state in which the above operations can be transferred.
Postcondition:	SSF FSM	any state as result of the transfer of any of the above operations.

b) *SSF receives Error "MissingParameter"*

Precondition:	SSF FSM	any state as result of the transfer of any of the above operations.
Postcondition:	SSF FSM state a	Idle.

After receiving this Error, the SSF FSM returns to the state Idle. The CCF routes the call if necessary (default routing to a terminating announcement). If the call is already established (i.e. mid-call trigger or ApplyChargingReport), the CCF may maintain the call or disconnect it. The choice between these two options is network-operator specific. In case of an assisting SSF, the temporary connection is released by the assisting SSF.

Procedures at responding entity (SCF)

Precondition:	1)	SCSM	appropriate state.
	2)	SCSM	Operation received, appropriate event occurred.
Postcondition:	1)	SCSM state 1	Idle; in case of any operation listed above, except AssistRequestInstructions; or
	2)	SCSM state 2.1	Preparing SSF Instructions; in case of AssistRequestInstructions.

The SCSM detects the erroneous situation. The Error parameter is used to inform the SSF of this situation. The Service Logic and maintenance functions are informed.

16.1.9.4 Operations SCF→SRF

Cancel
ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 3.1	Determine Mode; PromptAndCollectUserInformation or PlayAnnouncement will accompany the ConnectToResource; or
	SCSM state 3.2	Waiting for AssistRequestInstructions; after EstablishTemporaryConnection; or
	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformation are outstanding.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

b) *Receiving Error*

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

Error treatment depends on Service logic. SCF can initiate new User Interaction or force Disconnect (to SSF).

Procedures at responding entity (SRF)

Precondition:	SRSM state 2	Connected; or
	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

The SRSM detects that a required parameter is not present in the Operation argument. The Error parameter MissingParameter is used to inform the SCF of this situation. The SCF should take the appropriate actions to treat this error.

16.1.9.5 Operations SRF→SCF

AssistRequestInstructions

Procedures at invoking entity (SRF)

a) *Sending Operation*

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 2	Connected.

b) *Receiving Error*

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 1	Idle.

Procedures at responding entity (SCF)

Precondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

Postcondition: SCSM state 2.1 Preparing SSF instructions.

The SCSM detects the error in the received operation. The Error parameter is used to inform the SRF of this situation. The Service Logic and maintenance functions are informed. The SCF might try another SRF, route the call or release the call (Service Logic dependent).

16.1.9.6 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

a) *Sending operation*

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state Assisting Mode (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in Assisting Mode (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or

SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

b) *Receiving Error*

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or

SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state Idle (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Error treatments in controlling SCF or chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

a) *Receiving Operation*

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or

SCSM-Con in state Assisted Mode (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in Assisted Mode (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in Assisted Mode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

b) *Returning Error*

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided

or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Error treatments in controlling SCF or chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

16.1.9.7 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	FSM for CUSF state N1	Idle; or
	FSM for CUSF state N2.1	Preparing CUSF Instructions.
Postcondition:	FSM for CUSF state N2.1	Preparing CUSF Instructions; or
	FSM for CUSF state N2.2	Waiting for Notification or Request.

b) *SCF receives Error "MissingParameter"*

Precondition:	FSM for CUSF state N2.1	Preparing CUSF Instructions; or
	FSM for CUSF state N2.2	Waiting for Notification or Request.
Postcondition:	FSM for CUSF	transition to the initial state (i.e. before sending the erroneous operation).

The Service Logic and maintenance functions are informed. Further treatment of the call is dependent on Service Logic.

Procedures at responding entity (CUSF)

Precondition:	1)	CUSF FSM	any appropriate state.
	2)	CUSF FSM	an operation received, appropriate event occurred.
Postcondition:	1)	CUSF FSM	transition to the same state.

The CUSF FSM detects the error in the received operation. The Error parameter is returned to inform the SCF of this situation.

16.1.9.8 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

a) *Sending Operation*

Precondition:	CUSF FSM state a	Idle; or
	CUSF FSM state b	Waiting for Instructions; or
	CUSF FSM state c	Monitoring.

Postcondition:	CUSF FSM state a	Idle; or
	CUSF FSM state b	Waiting for Instructions; or
	CUSF FSM state c	Monitoring.

b) *CUSF receives Error "MissingParameter"*

Precondition:	CUSF FSM state a	Idle; or
	CUSF FSM state b	Waiting for Instructions; or
	CUSF FSM state c	Monitoring.
Postcondition:	CUSF FSM state a	Idle.

After receiving this Error, the CUSF FSM returns to the state Idle. The CUSF terminates the association if necessary. If the supplementary service is already active and ready for responding, the CUSF may maintain the association and continue service processing. The choice between these two options is network operator specific.

Procedures at responding entity (SCF)

Precondition:	1)	FSM for CUSF	any appropriate state.
	2)	FSM for CUSF	Operation received, appropriate event occurred.
Postcondition:	1)	FSM for CUSF state N1	Idle; in case of any operation listed above.

The FSM for CUSF detects the erroneous situation. The Error parameter is used to inform the CUSF of this situation. The Service Logic and maintenance functions are informed.

16.1.10 Name Error

16.1.10.1 General description

16.1.10.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to the name of the object. The conditions under which a name error is to be issued are defined in Recommendation X.511 (1993) subclause 12.5.

16.1.10.1.2 Argument description

The name error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.5.

16.1.10.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
RemoveEntry
Search

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.
Postcondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

The SDF could not perform the operation due to a name problem and therefore sends an Name error to the SCF. After returning the error, no further error treatment is performed.

16.1.10.3 Operations SDF→SDF

ChainedAddEntry
ChainedExecute
ChainedModifyEntry
ChainedRemoveEntry
ChainedSearch

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

This error is reported to the SCF.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

b) *Returning Error*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

The SDF could not perform the operation due to a name problem and therefore sends an Name error to the other SDF. After returning the error, no further error treatment is performed.

16.1.11 ParameterOutOfRange

16.1.11.1 General description

16.1.11.1.1 Error description

The responding entity cannot start the processing of the requested Operation because an Error in a parameter of the Operation argument is detected: a parameter value is out of range. This error is applied for the following two cases (when the error is determined by the application):

1) For the parameter which type is defined with the range of its size, such as INTEGER(x..y), SEQUENCE SIZE(x..y) OF Type. This error is applied when the parameter value is z or the parameter size is z where z < x or z > y.

2) For the parameter which type is defined as list of ENUMERATED value, the ParameterOutOfRange error is applied when the parameter value is not equal to any of the ENUMERATED values in the list.

16.1.11.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
SendChargingInformation
SendSTUI

Call associated/call processing

AnalyseInformation
CollectInformation
Connect
InitiateCallAttempt
RequestNotificationChargingEvent
RequestReportBCSMEvent
ResetTimer
SelectFacility
SelectRoute

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.11.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
InitialDP
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall

ONoAnswer
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.11.4 Operations SCF→SRF

PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.11.5 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.11.6 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.11.7 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.12 Referral

16.1.12.1 General description

16.1.12.1.1 Error description

This error is sent by the SDF to the SCF to report a problem related to service data location. The conditions under which a referral error is to be issued are defined in Recommendation X.511 (1993) subclause 12.6.

16.1.12.1.2 Argument description

The referral error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.6.

16.1.12.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
RemoveEntry
Search

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.
Postcondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

After receiving a referral error, the SCF can continue to execute the service logic by accessing another SDF which is indicated by this error.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

The SDF could not perform the operation due to a data location and therefore sends a Referral error to the SCF. After returning the error, no further error treatment is performed.

16.1.13 RequestedInfoError

16.1.13.1 General description

16.1.13.1.1 Error description

The RequestedInfoError is an immediate response to the CallInformationRequest operation, indicating that the requested information is not known to the SSF or is not available. RequestedInfoError is used when a specific SSF/CCF cannot offer the information specified with RequestedInformationType but there exists other SSF/CCF that can offer the information.

16.1.13.1.2 Argument description

PARAMETER ENUMERATED {
unknownRequestedInfo(1),
requestedInfoNotAvailable(2)
-- other values not specified
}

16.1.13.2 Operations SCF→SSF

CallInformationRequest

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.14 ScfReferral

16.1.14.1 General description

16.1.14.1.1 Error description

This error is sent by the supporting SCF to the controlling SCF or by the chaining terminator to the chaining initiator, in order to report that it is not able to provide the assistance and to provide the address of another SCF.

16.1.14.1.2 Argument description

The scfReferral error is defined through the AccessPointInformation parameter specified in Recommendation X.511 (1993).

16.1.14.2 SCF-SCF operation

ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest

ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification
HandlingInformationRequest

Procedures at invoking entity (controlling SCF or supporting chaining initiator SCF)

a) *Sending Operation*

Precondition: SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest; or
SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest.

Postcondition: SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest.

b) *Receiving Error*

Precondition: SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest.

Postcondition: SCSM-Con in state Idle (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest.

After receiving the SCFreferral error, the SCF can request assistance to another SCF.

Procedures at responding entity (SCF)

a) *Receiving Operation*

Precondition: SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest); or

SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest).

Postcondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest).

b) *Returning Error*

Precondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest).

Postcondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest).

After returning the error, the assisting SCF is waiting for SCFUnbind operation. No further error treatments take place. If a guard timer expires, event (e22) in SCSM-Sup or (e6) in SCSM-ChT will occur.

16.1.15 Security

16.1.15.1 General description

16.1.15.1.1 Error description

This error is sent by the SCF/SDF to the SCF/SDF to report a problem in carrying out an operation for security reasons. The conditions under which a security error is to be issued are defined in Recommendation X.511 (1993) subclause 12.7.

16.1.15.1.2 Argument description

The security error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.7.

16.1.15.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
RemoveEntry
Search

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.

b) *Receiving Error*

Precondition:	SCSM state 3	Wait for Bind result; or
	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 1	Idle; or
	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 2	Bind Pending; or
	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 2	Bind Pending; or
	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound

The SDF could not perform the operation for security reasons and therefore sends a Security error to the SCF. After returning the error, no further error treatment is performed.

16.1.15.3 Operations SDF→SDF

ChainedAddEntry
ChainedExecute
ChainedModifyEntry
ChainedRemoveEntry
ChainedSearch

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition: SDSM-ChI state 4 SDF Bound.

Postcondition: SDSM-ChI state 4 SDF Bound.

This error is reported to the SCF in case of chained operations.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition: SDSM-ChT state 3 SDF Bound.

Postcondition: SDSM-ChT state 3 SDF Bound.

b) *Returning Error*

Precondition: SDSM-ChT state 3 SDF Bound.

Postcondition: SDSM-ChT state 3 SDF Bound.

The SDF could not perform the operation for security reasons and therefore sends a Security error to the other SDF. After returning the error, no further error treatment is performed.

16.1.15.4 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition: SCSM-Con in state Idle (in case of SCFBind); or
SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state Assisting Mode (in case of NetworkCapability)

SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in Assisting Mode (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChI in state Idle (in case of SCFBind used in the chaining); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state PreparingRequestForAssistance (in case of SCFBind); or
SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification); or
SCSM-ChI in state PrepareChainedHIReq (in case of SCFBind used in the chaining); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

b) *Receiving Error*

Precondition: SCSM-Con in state PreparingRequestForAssistance (in case of SCFBind); or
SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord); or
SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest); or
SCSM-Sup in state Assisting mode (in case of HandlingInformationResult); or
SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability); or
SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Con in Assisted Mode (in case of ReportChargingInformation); or
SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state PrepareChainedHIReq (in case of SCFBind used in the chaining); or
SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Con in state Idle (in case of SCFBind)
or SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Idle (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state Idle (in case of SCFBind used in the chaining)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state Idle (in case of first

ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Once the security error is reported to the controlling SCF or chaining initiator internal logic, the need to send a SCFUnbind operation is determined.

Once the supporting SCF or the chaining terminator SCF receives a security error, it expects to receive an SCFUnbind operation. This event is guarded by a expiration timer.

Procedures at responding entity (SCF)

a) *Receiving Operation*

Precondition: SCSM-Sup in state Idle (in case of SCFBind); or
SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state Assisted Mode (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in Assisted Mode (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in Assisted Mode (in case of RequestNotification); or
SCSM-ChT in state Idle (in case of SCFBind used in the chaining); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state SCFBindRequestReceived (in case of SCFBind); or
SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest); or

or SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChT in state BindPending (in case of SCFBind used in the chaining); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

b) *Returning Error*

Precondition: SCSM-Sup in state SCFBindRequestReceived (in case of SCFBind); or
SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChT in state BindPending (in case of SCFBind used in the chaining); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or

SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-Sup in state Idle (in case of SCFBind); or
SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord); or
SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest); or
SCSM-Con in state Assisted mode (in case of HandlingInformationResult); or
SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability); or
SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest); or
SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation); or
SCSM-Sup in Assisting Mode (in case of ReportChargingInformation); or
SCSM-Con in AssistedMode (in case of RequestNotification); or
SCSM-ChT in state Idle (in case of SCFBind used in the chaining); or
SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

The security error is reported to the controlling SCF internal logic and the need to send a SCFUnbind operation is determined.

Once the supporting SCF or the chaining terminator SCF receives a security error, if this event does not cause a transition to idle, it expects to receive an SCFUnbind operation. This event is guarded by a expiration timer.

16.1.16 Service

16.1.16.1 General description

16.1.16.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to the provision of the service. The conditions under which a service error is to be issued are defined in Recommendation X.511 (1993) subclause 12.8.

16.1.16.1.2 Argument description

The Service error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.8.

16.1.16.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
RemoveEntry
Search

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 2	Wait for subsequent requests; or
	SCSM state 4	SDF Bound.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic. If there is an alternative SDF it may be possible to do another query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 2	Bind Pending (in case of Bind); or
	SDF FSM state 3	SCF Bound (in case of operations except Bind).
Postcondition:	SDF FSM state 1	Idle (in case of Bind); or
	SDF FSM state 3	SCF Bound (in case of operations except Bind).

The SDF could not perform the operation due to a service related problem and sends a Service error to the SCF. After returning the error, no further error treatment is performed.

16.1.16.3 Operations SDF→SDF

ChainedAddEntry
ChainedExecute

ChainedModifyEntry
ChainedRemoveEntry
ChainedSearch

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

This error is reported to the SCF in case of chained operations.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

b) *Returning Error*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

The SDF could not perform the operation due to a service related problem and sends a Service error to the other SDF. After returning the error, no further error treatment is performed.

16.1.17 Shadow

16.1.17.1 General description

16.1.17.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to shadowing. The conditions under which a shadow error is to be issued are defined in Recommendation X.525 (1993) clause 12.

16.1.17.1.2 Argument description

The shadow error parameter and problem codes are specified in Recommendation X.525 (1993) subclause 11.3.3.

16.1.17.2 Operations SDF→SDF

CoordinateShadowUpdate
RequestShadowUpdate
UpdateShadow

Procedures at Supplier Entity (SDF)

a-1) *Sending Operation*

Precondition: SDSM-ShM (in case of supplier initiated)
state 4 SDF Bound (in case of CoordinateShadowUpdate); or
state 6 Wait for update (in case of UpdateShadow).
SDSM-ShM (in case of consumer initiated).
state 5 Wait for update (in case of UpdateShadow).

Postcondition: SDSM-ShM (in case of supplier initiated).
state 5 Wait for coordination result (in case of CoordinateShadowUpdate); or
state 7 Wait for update confirmation (in case of UpdateShadow).
SDSM-ShM (in case of consumer initiated).
state 6 Wait for update confirmation (in case of UpdateShadow).

b-1) *Receiving Error*

Precondition: SDSM-ShM (in case of supplier initiated).
state 5 Wait for coordination result (in case of CoordinateShadowUpdate); or
state 7 Wait for update confirmation (in case of UpdateShadow).
SDSM-ShM (in case of consumer initiated).

Postcondition: SDSM-ShM (in case of supplier initiated).
state 4 SDF Bound (in case of CoordinateShadowUpdate); or
state 6 Wait for update (in case of UpdateShadow).
SDSM-ShM (in case of consumer initiated).
state 5 Wait for update (in case of UpdateShadow).

a-2) *Receiving Operation*

Precondition: SDSM-ShM (in case of consumer initiated).

Postcondition: SDSM-ShM (in case of consumer initiated).
state 4 Wait for RequestShadow result (in case of RequestShadowUpdate).

b-2) *Returning Error*

Precondition: SDSM-ShM (in case of consumer initiated).
state 4 Wait for RequestShadow result (in case of RequestShadowUpdate).

Postcondition: SDSM-ShM (in case of consumer initiated).
state 3 SDF Bound (in case of RequestShadowUpdate).

After receiving or returning the error, no further error treatment is performed.

Procedures at Consumer Entity (SDF)

a-1) *Sending Operation*

Precondition: SDSM-ShC (in case of consumer initiated).
state 4 SDF Bound (in case of RequestShadowUpdate).

Postcondition: SDSM-ShC (in case of consumer initiated).
state 5 Wait for RequestShadow result (in case of RequestShadowUpdate).

b-1) *Receiving Error*

Precondition: SDSM-ShC (in case of RequestShadowUpdate)
state 5 Wait for RequestShadow result (in case of RequestShadowUpdate).

Postcondition: SDSM-ShC (in case of RequestShadowUpdate).
state 4 SDF Bound (in case of RequestShadowUpdate).

a-2) *Receiving Operation*

Precondition: SDSM-ShC (in case of supplier initiated).
state 3 SDF Bound (in case of CoordinateShadowUpdate); or
state 5 Wait for update (in case of UpdateShadow).
SDSM-ShC (in case of consumer initiated).
state 6 Wait for update (in case of UpdateShadow).

Postcondition: SDSM-ShC (in case of supplier initiated).
state 4 Wait for coordination result (in case of CoordinateShadowUPdate);or
state 6 Wait for update confirmation (in case of UpdateShadow).
SDSM-ShC (in case of consumer initiated).
state 7 Wait for update confirmation (in case of UpdateShadow).

b-2) *Returning Error*

Precondition: SDSM-ShC (in case of supplier initiated).
state 4 Wait for coordination result (in case of CoordinateShadowUPdate); or
state 6 Wait for update confirmation (in case of UpdateShadow).
SDSM-ShC (in case of consumer initiated).
state 7 Wait for update confirmation (in case of UpdateShadow).

Postcondition: SDSM-ShC (in case of supplier initiated).
state 3 SDF Bound (in case of CoordinateShadowUpdate); or
state 5 Wait for update (in case of UpdateShadow).
SDSM-ShC (in case of consumer initiated).
state 6 Wait for update (in case of UpdateShadow).

After receiving or returning the error, no further error treatment is performed.

16.1.18 SystemFailure

16.1.18.1 General description

16.1.18.1.1 Error description

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

16.1.18.1.2 Argument description

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

16.1.18.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent
RequestReportFacilityEvent
RequestReportUTSI
SendChargingInformation
SendFacilityInformation
SendSTUI

Call associated/call processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
CreateCallSegmentAssociation
DisconnectForwardConnection
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments
MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer

OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
Osuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
Tanswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer
TSuspended

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.4 Operations SCF→SRF

ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.5 Operations SRF→SCF

AssistRequestInstructions

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.6 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability

ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.7 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.18.8 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19 TaskRefused

16.1.19.1 General introduction

16.1.19.1.1 Error description

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

16.1.19.1.2 Argument description

```
PARAMETER ENUMERATED {
        generic(0),
        unobtainable (1),
        congestion (2)
          }
```

16.1.19.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
Cancel
CancelStatusReportRequest
FurnishChargingInformation
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent
RequestReportFacilityEvent
RequestReportUTSI
ResetTimer
SendChargingInformation
SendFacilityInformation
SendSTUI

Call associated/call processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
CreateCallSegmentAssociation
DisconnectForwardConnection
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments
MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
AssistRequestInstructions

CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer
TSuspended

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19.4 Operations SCF→SRF

Cancel
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptClose
ScriptInformation
ScriptRun

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19.5 Operations SRF→SCF

AssistRequestInstructions

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19.6 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.19.7 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.20 UnavailableResource

16.1.20.1 General description

16.1.20.1.1 Error description

The SRF is not able to perform its function (i.e. play a certain announcement and/or collect specific user information), and cannot be replaced. A re-attempt is not possible.

16.1.20.2 Operations SCF→SRF

ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

Procedures at invoking entity (SCF)

a) *SCF sends PlayAnnouncement or PromptAndCollectUserInformation to SRF*

Precondition:	SCSM state 3.1	Determine Mode; PlayAnnouncement or PromptAndCollectUserInformation will accompany the ConnectToResource; or
	SCSM state 3.2	Waiting for AssistRequestInstructions; after EstablishTemporaryConnection; or
	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformation are outstanding.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

b) *SCF receives UnavailableResource Error from SRF*

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

If the chosen resource cannot perform its function, the further treatment is service dependent.

Examples:

– request SSF to connect to alternative SRF;
– service processing without PlayAnnouncement or PromptAndCollectUserInformation (if possible);

– terminate service processing.

Procedures at responding entity (SRF)

a) *SRF receiving PlayAnnouncement or PromptAndCollectUserInformation*

Precondition:	SRSM state 2	Connected; if initial PlayAnnouncement or PromptAndCollectUserInformation; or
	SRSM state 3	User Interaction; if not initial PlayAnnouncement or PromptAndCollectUserInformation.

b) *SRF is not able to perform its function (and cannot be replaced). SRF sends UnavailableResource.*

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

16.1.21 UnexpectedComponentSequence

16.1.21.1 General description

16.1.21.1.1 Error description

The responding entity cannot start the processing of the requested operation because a SACF or MACF rule is violated, or the operation could not be processed in the current state of the FSM.

16.1.21.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
FurnishChargingInformation
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent
RequestReportFacilityEvent
RequestReportUTSI
ResetTimer
SendChargingInformation
SendFacilityInformation
SendSTUI

Call Associated/Call Processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
ContinueWithArgument

CreateCallSegmentAssociation
DisconnectForwardConnection
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments
MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

In this case, the SSF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the Service Logic and maintenance functions are informed and the Service Logic decides about error treatment.

16.1.21.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
AssistRequestInstructions
CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer
TSuspended

In case of assisting SSF an error occurs in case an AssistRequestInstructions is sent while a relationship between SCF and assisting SSF has already been established, the SCF returns the error parameter. Service Logic and maintenance are informed. On receiving the error, the assisting SSF moves to Idle and the temporary connection is released.

In case the operation is sent by an "initiating" SSF in the context of an existing relationship, the SCF returns the error parameter. Service Logic and maintenance are informed. On receiving the error, the SSF moves to Idle.

16.1.21.4 Operations SCF→SRF (only applicable for direct SCF-SRF case)

ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

In this case the SRF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the Service Logic and maintenance functions are informed and the Service Logic decides about error treatment. Possible error treatment is to send the DisconnectForwardConnection operation to the SSF.

16.1.21.5 Operations SRF→SCF

AssistRequestInstructions

In this case, an error occurs if the SRF has already an established relationship with the SCF and sends an AssistRequestInstructions. The SCF detects the erroneous situation, informs Service Logic and maintenance functions and returns the error parameter. On receiving the parameter, the SRF moves to idle and releases the temporary connection.

16.1.21.6 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.21.7 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF) and responding entity (CUSF)

The CUSF detects the error, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the Service Logic and maintenance functions are informed and the Service Logic determines the appropriate error treatment.

16.1.21.8 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF) and responding entity (SCF)

In case the operation is sent by CUSF, the SCF returns the error parameter. Service Logic and maintenance functions are informed, and the Service Logic determines the appropriate error treatment. On receiving the error the CUSF moves to Idle.

16.1.22 UnexpectedDataValue

16.1.22.1 General description

16.1.22.1.1 Error description

The responding entity cannot complete the processing of the requested Operation because a parameter has an unexpected data value.

Note that this error does not overlap with "ParameterOutOfRange"

Example: startTime DateAndTime ::= -- *value indicating January 32 1993, 12:15:01*

The responding entity does not expect this value and responds with "UnexpectedDataValue".

16.1.22.2 Operations SCF→SSF

Non-call associated

ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
FurnishChargingInformation
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent

RequestReportFacilityEvent
RequestReportUTSI
ResetTimer
SendFacilityInformation
SendSTUI

Call associated/call processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
ContinueWithArgument
CreateCallSegmentAssociation
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments
MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
AssistRequestInstructions
CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized

TMidCall
TNoAnswer
TSuspended

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.4 Operations SCF→SRF

ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.5 Operations SRF→SCF

AssistRequestInstructions

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.6 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.7 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.22.8 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23 UnexpectedParameter

16.1.23.1 General description

16.1.23.1.1 Error description

There is an error in the received Operation argument. A valid but unexpected parameter was present in the Operation argument. The presence of this parameter is not consistent with the presence of the other parameters. The responding entity cannot start to process the Operation.

16.1.23.2 Operations SCF→SSF

Non-call associated

ActivateServiceFiltering
ManageTriggerData

Call associated/non-call processing

ApplyCharging
CallInformationRequest
FurnishChargingInformation
RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport
RequestNotificationChargingEvent
RequestReportBCSMEvent
RequestReportFacilityEvent
RequestReportUTSI
ResetTimer
SendChargingInformation

SendFacilityInformation
SendSTUI

Call associated/call processing

AnalyseInformation
AuthorizeTermination
CollectInformation
Connect
ConnectToResource
ContinueWithArgument
CreateCallSegmentAssociation
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
HoldCallInNetwork
InitiateCallAttempt
MergeCallSegments
MoveCallSegments
MoveLeg
Reconnect
SelectFacility
SelectRoute
SplitLeg

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.3 Operations SSF→SCF

AnalysedInformation
ApplyChargingReport
AssistRequestInstructions
CollectedInformation
FacilitySelectedAndAvailable
InitialDP
OAbandon
OAnswer
OCalledPartyBusy
ODisconnect
OMidCall
ONoAnswer
OSuspended
OriginationAttempt
OriginationAttemptAuthorized
RouteSelectFailure
TAnswer
TBusy
TDisconnect
TerminationAttempt
TermAttemptAuthorized
TMidCall
TNoAnswer
TSuspended

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.4 Operations SCF→SRF

ScriptClose
ScriptInformation
PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
ScriptRun

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.5 Operations SRF→SCF

AssistRequestInstructions

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.6 Operations SCF→SCF

ConfirmedNotificationProvided
ConfirmedReportChargingInformation
EstablishChargingRecord
HandlingInformationRequest
HandlingInformationResult
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedConfirmedNotificationProvided
ChainedConfirmedReportChargingInformation
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.7 Operations SCF→CUSF

InitiateAssociation
RequestReportBCUSMEvent
SendComponent

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.23.8 Operations CUSF→SCF

ActivationReceivedAndAuthorized
AssociationReleaseRequested
ComponentReceived

Procedures at invoking entity (CUSF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.24 UnknownLegID

16.1.24.1 General description

16.1.24.1.1 Error description

This error is used to indicate to the SCF that a specific leg, indicated by the LegID parameter value in the operation, is unknown to the SSF.

16.1.24.2 Operations SCF→SSF

Call associated/non-call processing

ApplyCharging
CallInformationRequest
RequestReportFacilityEvent
SendChargingInformation
SendFacilityInformation
SendSTUI

Call associated/call processing

ConnectToResource
DisconnectForwardConnectionWithArgument
DisconnectLeg
EstablishTemporaryConnection
InitiateCallAttempt
MoveCallSegments
MoveLeg
OAbandon
OSuspended
SplitLeg

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.25 UnknownResource

16.1.25.1 General description

16.1.25.1.1 Error description

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

16.1.25.2 Operations SCF→SSF

Call associated/non-call processing

RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.1.26 Update

16.1.26.1 General description

16.1.26.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to attempts to add, delete, or modify information in the SDF. The conditions under which an update error is to be issued are defined in Recommendation X.511 (1993) subclause 12.9.

16.1.26.1.2 Argument description

The update error parameter and problem codes are specified in Recommendation X.511 (1993) subclause 12.9.

16.1.26.2 Operations SCF→SDF

AddEntry
Execute
ModifyEntry
RemoveEntry

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.
Postcondition:	SCSM state 4	SDF Bound; or
	SCSM state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 3	SCF Bound.

Postcondition:	SDF FSM state 3	SCF Bound.

b) *Returning Error*

Precondition:	SDF FSM state 3	SCF Bound.
Postcondition:	SDF FSM state 3	SCF Bound.

The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the SCF. After returning the error, no further error treatment is performed.

16.1.26.3 Operations SDF→SDF

ChainedAddEntry
ChainedExecute
ChainedModifyEntry
ChainedRemoveEntry

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.
Postcondition:	SDSM-ChI state 4	SDF Bound; or
	SDSM-ChI state 2	Wait for subsequent requests.

b) *Receiving Error*

Precondition:	SDSM-ChI state 4	SDF Bound.
Postcondition:	SDSM-ChI state 4	SDF Bound.

This error is reported to the SCF.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

b) *Returning Error*

Precondition:	SDSM-ChT state 3	SDF Bound.
Postcondition:	SDSM-ChT state 3	SDF Bound.

The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the other SDF. After returning the error, no further error treatment is performed.

16.1.27 ChainingRefused

16.1.27.1 General description

16.1.27.1.1 Error description

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

16.1.27.1.2 Argument description

No parameters.

16.1.27.2 Operations SCF→SCF

ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedHandlingInformationResult
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-Con in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-Sup in SCFBound (in case of ChainedRequestNotification).

b) *Receiving Error*

Precondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or

SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation);
or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest); or
SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation);
or
SCSM-ChT in SCFBound (in case of ChainedRequestNotification).

Error treatments in chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

a) *Receiving Operation*

Precondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation);
or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation);
or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

b) *Returning Error*

Precondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Postcondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord); or
SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest); or
SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult); or
SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability); or
SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest); or
SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation); or
SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation); or
SCSM-ChI in SCFBound (in case of ChainedRequestNotification).

Error treatments in the chaining Initiator SCF depend on service logic. No further error treatment is performed in supporting SCF of chaining terminator SCF.

16.1.28 DirectoryBindError

16.1.28.1 General description

16.1.28.1.1 Error description

This error is sent by the SDF to the SCF/SDF to report a problem in establishing an authorized relationship. The conditions under which a directoryBindError is to be issued are defined in Recommendation X.511 (1997).

16.1.28.1.2 Argument description

The directoryBindError parameter and problem codes are specified in Recommendation X.511 (1997).

16.1.28.2 Operations SCF→SDF

directoryBind

Procedures at invoking entity (SCF)

a) *Sending Operation*

Precondition:	SCSM state 1	Idle.
Postcondition:	SCSM state 3	Wait for Bind result.

b) *Receiving Error*

Precondition:	SCSM state 3	Wait for Bind result.
Postcondition:	SCSM state 1	Idle.

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDF FSM state 1	Idle.
Postcondition:	SDF FSM state 2	Bind Pending.

b) *Returning Error*

Precondition:	SDF FSM state 2	Bind Pending (in case of Bind).
Postcondition:	SDF FSM state 1	Idle (in case of Bind).

The SDF could not perform the operation and therefore sends a directoryBindError to the SCF. After returning the error, no further error treatment is performed.

16.1.28.3 Operations SDF→SDF

DSABind
DSAShadowBind

Procedures at invoking entity (SDF)

a) *Sending Operation*

Precondition:	SDSM-ChI state 1	Idle (in case of DSABind); or
	SDSM-ShM state 1	Idle (in case of supplier initiated DSAShadowBind); or
	SDSM-ShC state 1	Idle (in case of consumer initiated DSAShadowBind).
Postcondition:	SDSM-ChI state 2	Wait for subsequent requests (in case of DSABind); or
	SDSM-ShM state 2	Wait for subsequent requests (in case of supplier initiated DSAShadowBind); or
	SDSM-ShC state 2	Wait for subsequent requests (in case of consumer initiated DSAShadowBind).

b) *Receiving Error*

Precondition:	SDSM-ChI state 3	Wait for Bind result (in case of DSABind); or
	SDSM-ShM state 3	Wait for Bind result (in case of supplier initiated DSAShadowBind); or

	SDSM-ShC state 3	Wait for Bind result (in case of consumer initiated DSAShadowBind).
Postcondition:	SDSM-ChI state 1	Idle (in case of DSABind); or
	SDSM-ShM state 1	Idle (in case of supplier initiated DSAShadowBind); or
	SDSM-ShC state 1	Idle (in case of consumer initiated DSAShadowBind).

This error is reported to the SCF in case of DSABind operations.

Procedures at responding entity (SDF)

a) *Receiving Operation*

Precondition:	SDSM-ChT state 1	Idle (in case of DSABind); or
	SDSM-ShC state 1	Idle (in case of supplier initiated DSAShadowBind); or
	SDSM-ShM state 1	Idle (in case of consumer initiated DSAShadowBind).
Postcondition:	SDSM-ChT state 2	Bind pending (in case of DSABind); or
	SDSM-ShC state 2	Wait for Bind result (in case of supplier initiated DSAShadowBind); or
	SDSM-ShM state 2	Wait for Bind result (in case of consumer initiated DSAShadowBind).

b) *Returning Error*

Precondition:	SDSM-ChT state 2	Bind pending (in case of DSABind); or
	SDSM-ShC state 2	Wait for Bind result (in case of supplier initiated DSAShadowBind); or
	SDSM-ShM state 2	Wait for Bind result (in case of consumer initiated DSAShadowBind).
Postcondition:	SDSM-ChT state 1	Idle (in case of DSABind); or
	SDSM-ShC state 1	Idle (in case of supplier initiated DSAShadowBind); or
	SDSM-ShM state 1	Idle (in case of consumer initiated DSAShadowBind).

The SDF could not perform the operations for security reasons and therefore sends a directoryBindError to the other SDF. In the case of DSABind, the error is reported to the SCF.

16.1.29 ScfBindFailure

16.1.29.1 General description

16.1.29.1.1 Error description

This error is sent by the supporting SCF (or terminator supporting in the chaining case) to the controlling SCF (or initiator supporting SCF) to report a problem in establishing an authorized relationship.

16.1.29.1.2 Argument description

```
PARAMETER
        FailureReason
FailureReason ::= CHOICE {
        systemFailure       [0] UnavailableNetworkResource,
        scfTaskRefused      [1] ScfTaskRefusedParameter,
        securityError       [2] SET {
                                problem     [0] SecurityProblem,
                                spkmInfo    [1] SPKM-ERROR
                                }
        }
```

16.1.29.2 Operations Controlling SCF→Supporting SCF

scfBind

Procedure at Invoking Entity (Controlling SCF)

a) *Sending Operation*

Precondition:	SCSM-Con state 1	Idle.
Postcondition:	SCSM-Con state 3	Wait for Bind result.

b) *Receiving Error*

Precondition:	SCSM-Con state 3	Wait for Bind result.
Postcondition:	SCSM-Con state 1	Idle.

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedure at Responding Entity (Supporting SCF)

a) *Receiving Operation*

Precondition:	SCSM-Sup state 1	Idle.
Postcondition:	SCSM-Sup state 2	Bind Pending.

b) *Returning Error*

Precondition:	SCF FSM state 2	Bind Pending (in case of Bind).
Postcondition:	SCF FSM state 1	Idle (in case of Bind).

The SCF could not perform the operation and therefore sends a Security error to the SCF. After returning the error, no further error treatment is performed.

16.1.29.3 Operations Chaining Initiator Supporting SCF→Terminator Supporting SCF

scfBind

Procedure at Invoking Entity (Initiator Supporting SCF)

a) *Sending Operation*

Precondition:	SCSM-ChI state 1	Idle.
Postcondition:	SCSM-ChI state 3	Wait for Bind result.

b) *Receiving Error*

Precondition:	SCSM-ChI state 3	Wait for Bind result.
Postcondition:	SCSM-ChI state 1	Idle.

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedure at Responding Entity (Terminator Supporting SCF)

a) *Receiving Operation*

Precondition:	SCSM-ChT state 1	Idle.
Postcondition:	SCSM-ChT state 2	Bind Pending.

b) *Returning Error*

Precondition:	SCSM-ChT state 2	Bind Pending (in case of Bind).
Postcondition:	SCSM-ChT state 1	Idle (in case of Bind).

16.1.30 ScfTaskRefused

16.1.30.1 General introduction

16.1.30.1.1 Error description

This Error is returned by a physical entity if it was not able to fulfill a specific task as requested by an operation. It is identical to taskRefused when security protection is not activated.

16.1.30.1.2 Argument description

```
PARAMETER
        ScfTaskRefusedParameter
ScfTaskRefusedParameter ::= OPTIONALLY-PROTECTED { SEQUENCE {
                reason          ENUMERATED {
                        generic(0),
                        unobtainable (1),
                        congestion(2)
                        },
                securityParameters [1] SecurityParameters      OPTIONAL
        },
        SCFQOP.&scfErrorsQOP{@scfqop}
        }
```

16.1.30.2 Operations SCF→SCF

EstablishChargingRecord
HandlingInformationRequest
NetworkCapability
NotificationProvided
ProvideUserInformation
ReportChargingInformation
RequestNotification
ChainedEstablishChargingRecord
ChainedHandlingInformationRequest
ChainedNetworkCapability
ChainedNotificationProvided
ChainedProvideUserInformation
ChainedReportChargingInformation
ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to 16.1.9, MissingParameter, for the appropriate error procedures.

16.2 Entity related error procedures

The following subclauses define the error handling for the entity related errors. Since the error situations are not originated by the reception of an operation, the invoking entity is denoted here as the entity at which the error situation is detected. The responding entity is the entity which receives the error report.

The TCAP services used for reporting errors are described in clause 18.

16.2.1 Expiration of T_{SSF}

16.2.1.1 General description

16.2.1.1.1 Error description

A timeout occurred in the SSF on the response from the SCF.

16.2.1.2 Procedures SSF→SCF

Procedures at the invoking entity (SSF)

Timeout occurs in SSF on T_{SSF}.

Precondition:	SSF FSM state c	Waiting for instructions; or
	SSF FSM state d	Waiting for end of User Interaction; or
	SSF FSM state e	Waiting for end of Temporary connection.
Postcondition:	SSF FSM state a	Idle.

The SSF FSM aborts the dialogue and moves to the Idle state, the CCF routes the call if necessary (e.g. default routing to a terminating announcement). The abort is reported to the maintenance functions.

Procedures at the responding entity (SCF)

SCF receives a dialogue abort.

Precondition:	Any state.	
Postcondition:	SCSM state 1	Idle; if the abort is related to a SSF dialogue; or
	SCSM state 2	Preparing SSF instructions; if the abort is related to an assisting SSF dialogue.

The SCF releases all allocated resources and reports the abort to the maintenance functions, if the abort is received on a SSF dialogue. The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions, if the abort is received on an assisting SSF dialogue.

16.2.2 Expiration of T_{SRF}

16.2.2.1 General description

16.2.2.1.1 Error description

A timeout occurred in the SRF on the response from the SCF. This procedure concerns only the direct SCF-SRF case.

16.2.2.2 Procedures SRF→SCF

Procedures at the invoking entity (SRF)

Timeout occurs in SRF on T_{SRF}

Precondition:	SRSM state 2	Connected; or
	SRSM state 3	User Interaction.
Postcondition:	SRSM state 1	Idle.

The SRF aborts the dialogue and moves to the Idle state, all allocated resources are de-allocated. The abort is reported to the maintenance functions.

Procedures at the responding entity (SCF)

SCF receives a dialogue abort.

Precondition:	SCSM state 4	User Interaction.
Postcondition:	SCSM state 2	Preparing SSF instructions.

The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions.

16.2.3 Expiration of T_{cusf}

16.2.3.1 General description

16.2.3.1.1 Error description

A timeout occurred in the CUSF on the response from the SCF.

16.2.3.2 Procedures CUSF→SCF

Procedures at the invoking entity (CUSF)

Timeout occurs in CUSF for T_{cusf}

Precondition:	CUSF FSM state b	Waiting for instructions.
Postcondition:	CUSF FSM state a	Idle.

The CUSF FSM aborts the dialogue and moves to the Idle state, the CUSF terminates the association if necessary (e.g. default exception handling). The abort is reported to the maintenance functions.

Procedures at the responding entity (SCF)

SCF receives a dialogue abort.

Precondition:	Any state.	
Postcondition:	FSM for CUSF state N1	Idle.

The SCF releases all allocated resources and reports the abort to the maintenance functions.

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 communication
Series Y	Global information infrastructure
Series Z	Programming languages