INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.1218

(03/93)

GENERAL RECOMMENDATIONS ON TELEPHONE SWITCHING AND SIGNALLING

INTELLIGENT NETWORK

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CS-1

ITU-T Recommendation Q.1218

(Previously "CCITT Recommendation")

FOREWORD

The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the International Telecommunication Union. 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, established the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

ITU-T Recommendation Q.1218 was prepared by the ITU-T Study Group XI (1988-1993) and was approved by the WTSC (Helsinki, March 1-12, 1993).

NOTES

1 As a consequence of a reform process within the International Telecommunication Union (ITU), the CCITT ceased to exist as of 28 February 1993. In its place, the ITU Telecommunication Standardization Sector (ITU-T) was created as of 1 March 1993. Similarly, in this reform process, the CCIR and the IFRB have been replaced by the Radiocommunication Sector.

In order not to delay publication of this Recommendation, no change has been made in the text to references containing the acronyms "CCITT, CCIR or IFRB" or their associated entities such as Plenary Assembly, Secretariat, etc. Future editions of this Recommendation will contain the proper terminology related to the new ITU structure.

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

CONTENTS

SUMMARY

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

There may be functional redundancies in the operation set related to call processing. This may make product interworking more difficult. Administrations wishing to deploy IN and equipment manufacturers implementing IN should take this into account.

It is expected that this Recommendation will have a number of enhancements made as implementation experience gained in the deployment of CS-1 brings omissions and errors to light. While these enhancements may be significant, they are unlikely to have large impact on existing text.

Appendix I contains work which indicates the direction in which future capability sets may evolve. This material has not had sufficient review to be considered stable and should be considered informative.

IN CS-1 is defined to minimize its impact on existing interexchange signalling and user-network signalling protocols including the decadic signalling systems being used in analogue networks. However, there exist some additional features required for them depending on the applications. How to implement these features is network dependent and is not specified in this Recommendation.

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

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CS-1

(Helsinki, 1993)

0 Introduction

This Recommendation defines the INAP (intelligent network application protocol) required for support of capability set 1. It supports interactions between the following four functional entities (FE's), as defined in the IN functional model:

- Service switching function (SSF)
- Service control function (SCF)
- Specialised resource function (SRF)
- Service data function (SDF)

0.1 Definition methodology

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

- the definition of the SACF/MACF rules for the protocol (Section 1);
- the definition of the operations transferred between entities (Section 2);
- the definition of the actions taken at each entity (Section 3).

The SACF/MACF rules are defined in prose. The operation definitions are in Abstract Syntax Notation 1 (ASN.1, see Recommendation X.208), and the actions are defined in terms of state transition diagrams. Further guidance on the actions to be performed on receipt of an operation can be gained from 2 and from the relevant information flow in 6/Q.1214 (see 0.5 for the relationship between the information flows and the operations).

The INAP is a ROSE user protocol (see Recommendaton X.219 and 229). The ROSE protocol is contained within the component sublayer of TCAP (see Recommendations Q.771 to 775) and DSS 1 (Recommendation Q.932). At present the ROSE APDU's (Application protocol data units) are conveyed in transaction sublayer messages in SS No. 7 and in the Q.931 REGISTER, FACILITY and call control messages in DSS 1. Other supporting protocols may be added at a later date.

Note that the TCAP component sublayer has an additional APDU Return Result Not Last which is not present in ROSE. This may be used only if *Blue Book* (1988) SCCP is used to support INAP.

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

0.2 Example physical scenarios

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

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

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

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

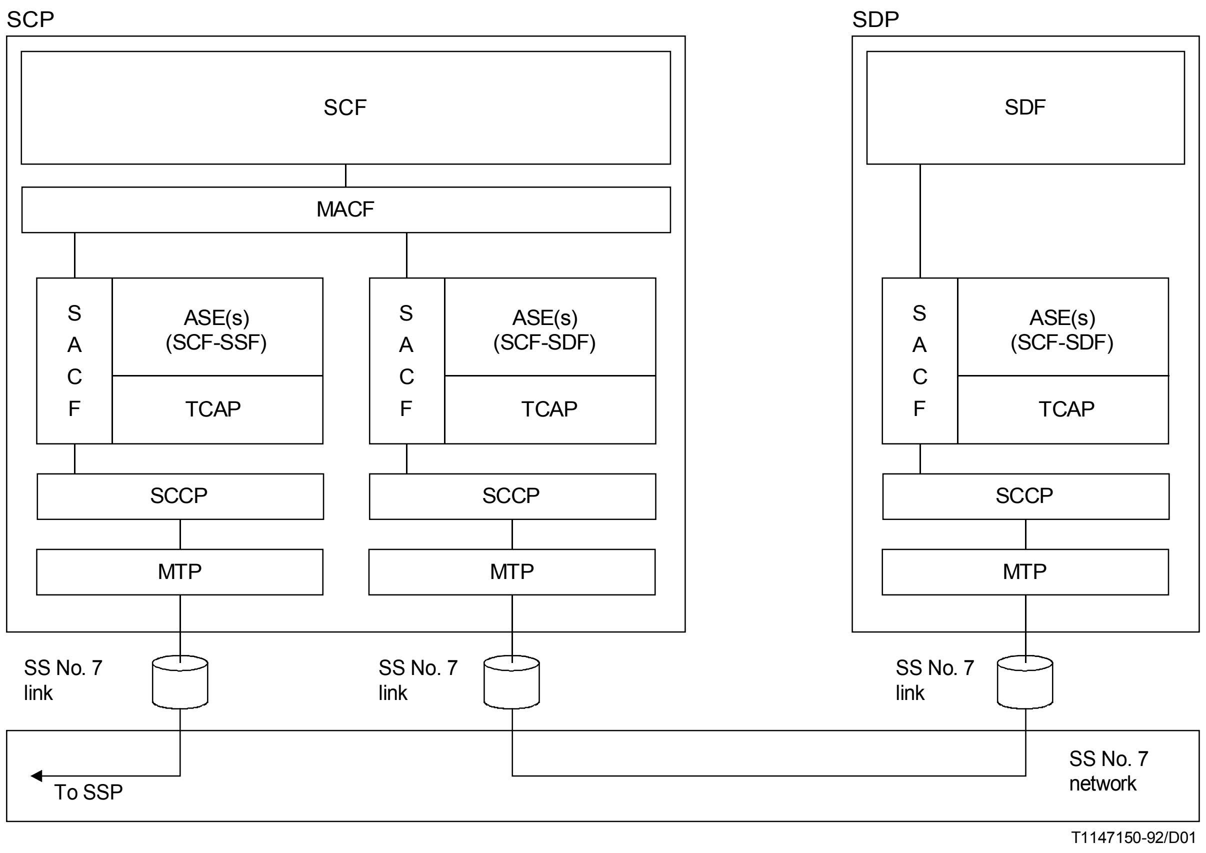


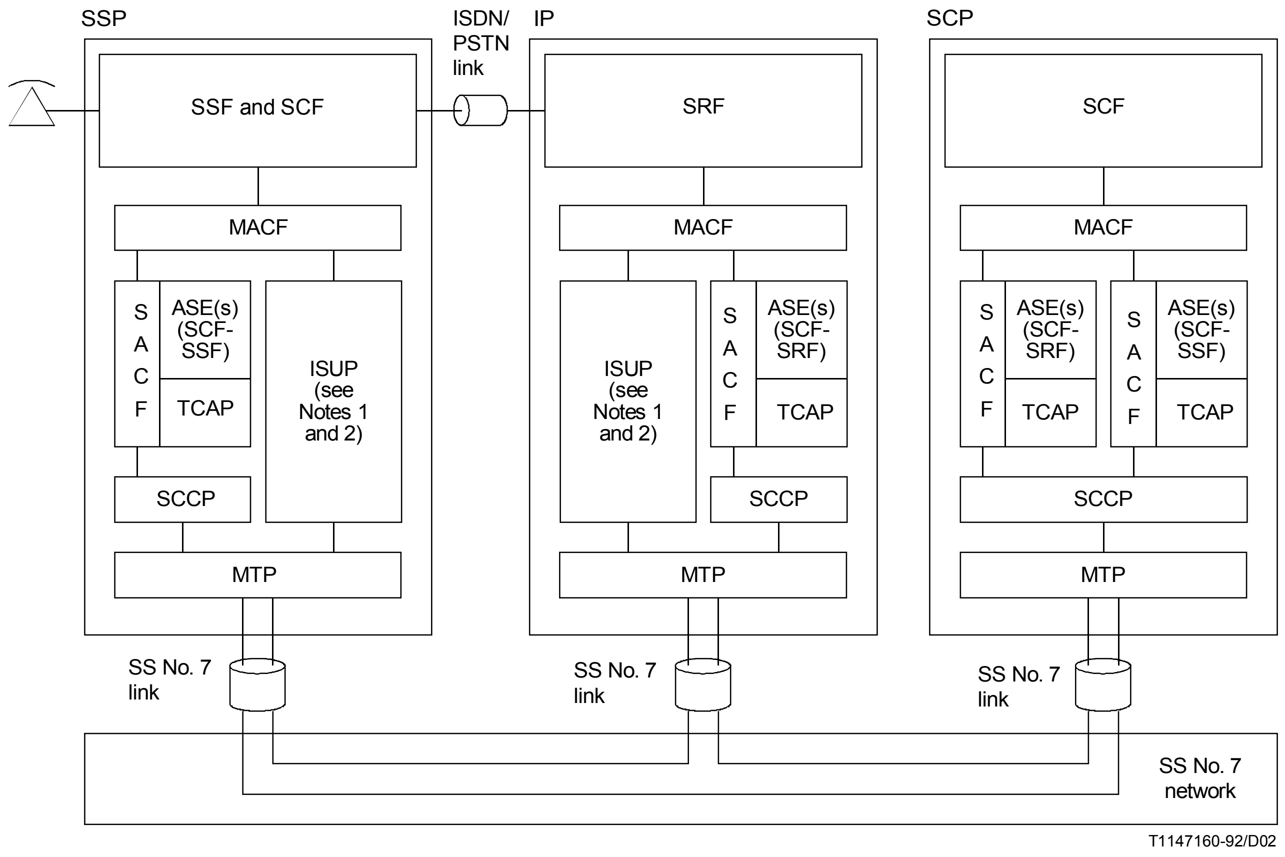
FIGURE 1/Q.1218

Physical interface between SCP and SDP

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

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

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



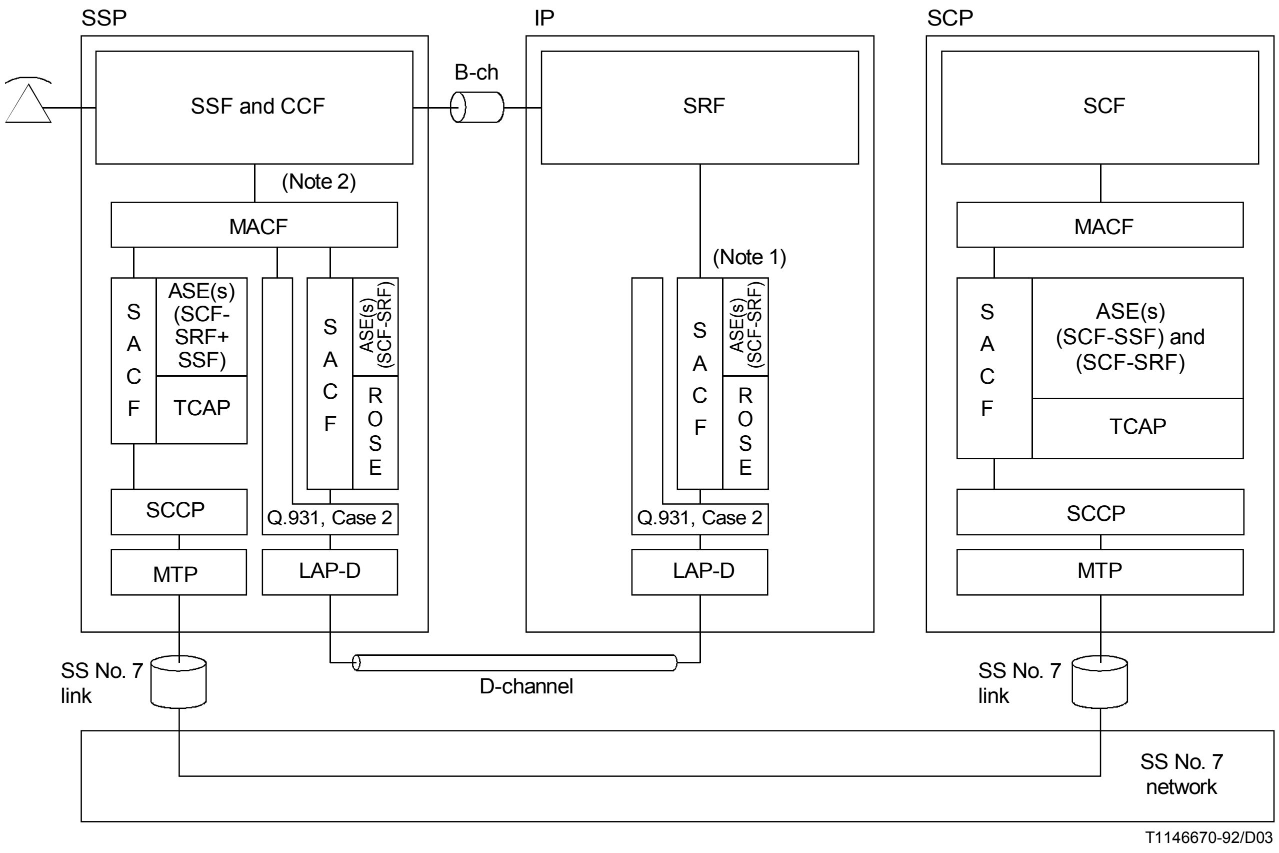
NOTES

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

2 Other signalling systems may be used.

FIGURE 2/Q.1218

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



NOTES

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

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

FIGURE 3/Q.1218

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

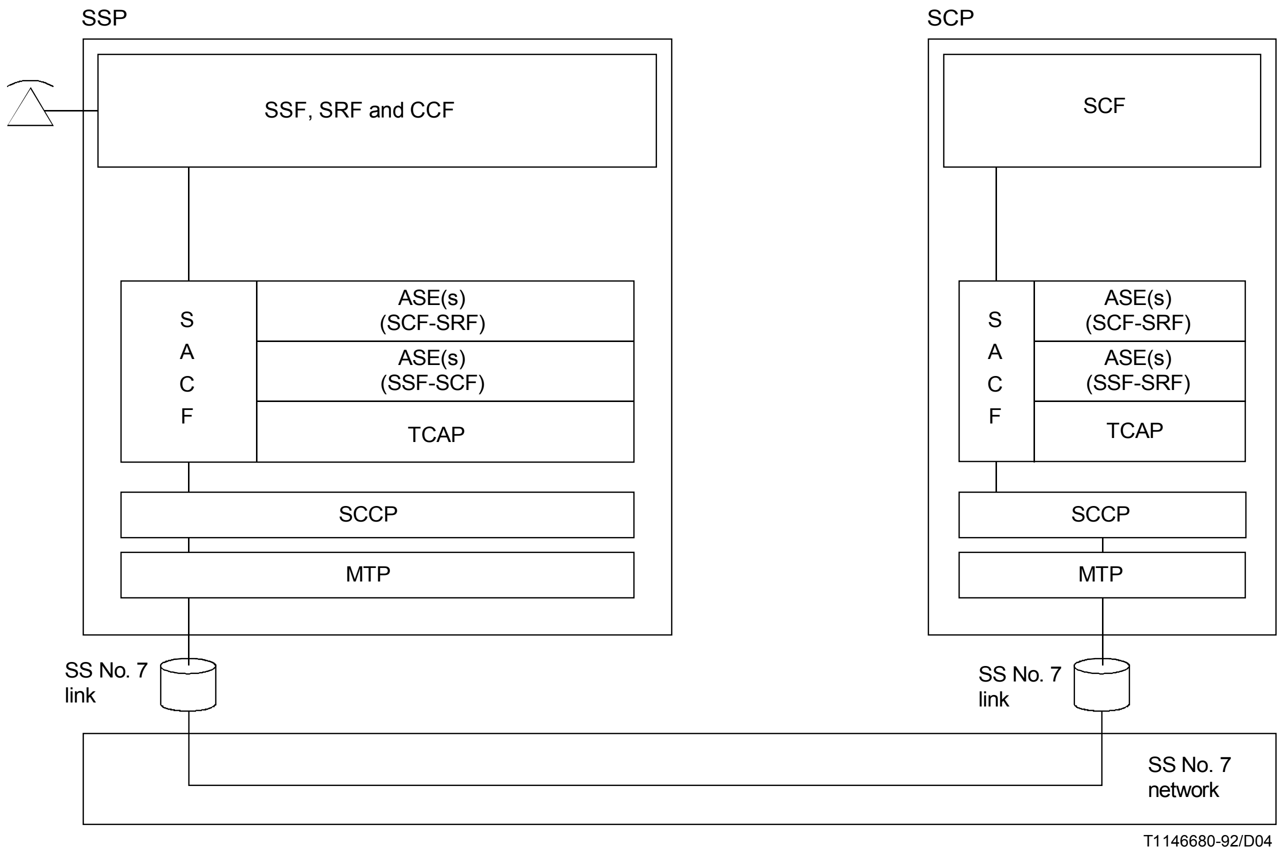
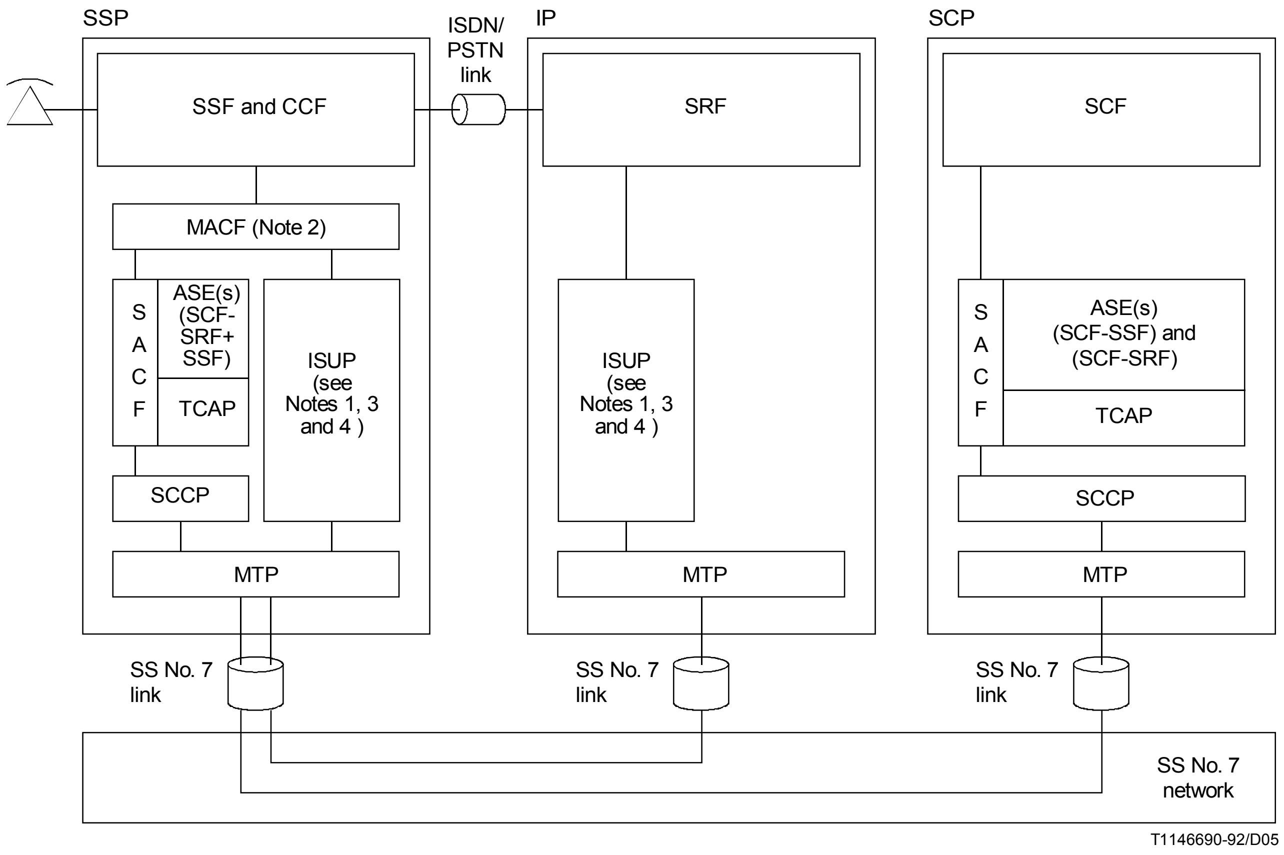


FIGURE 4/Q.1218

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



NOTES

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

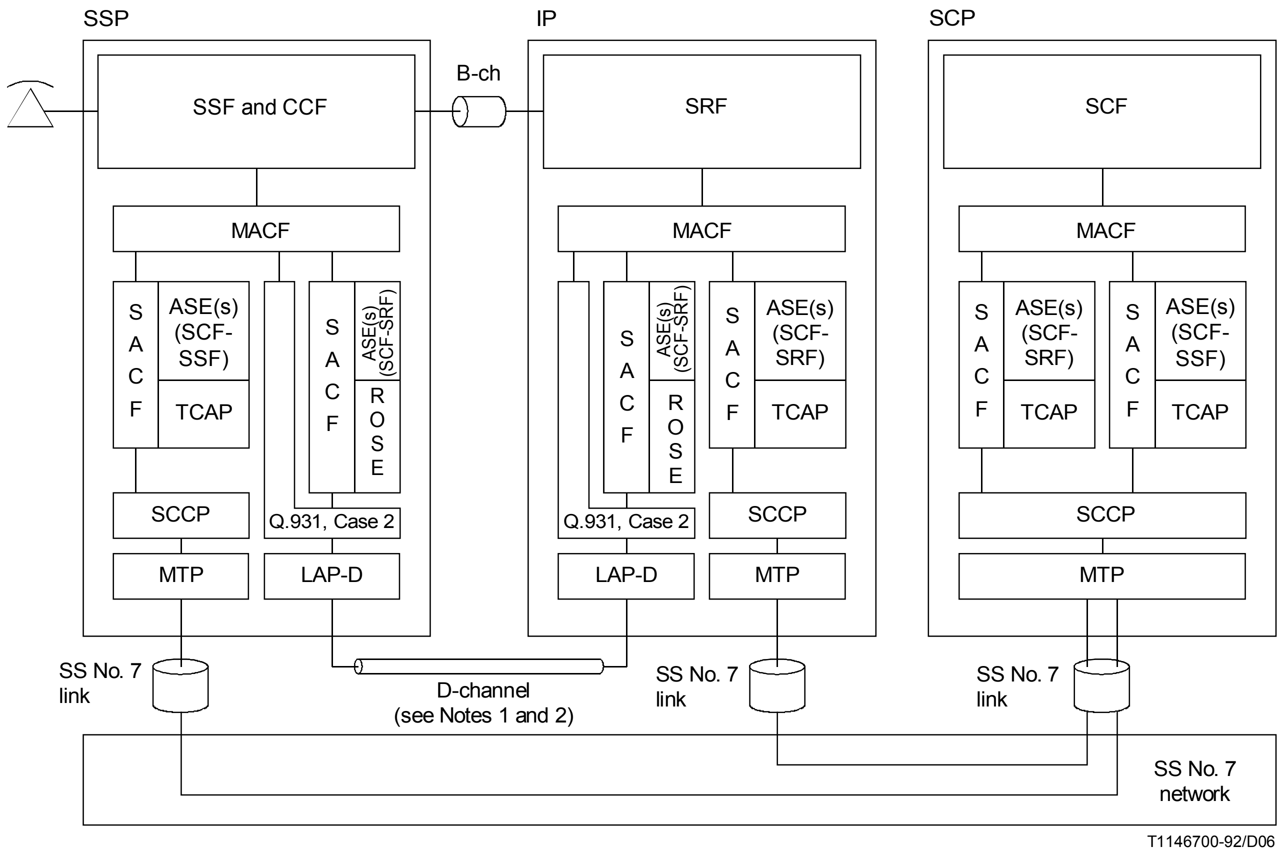
2 Relay function is provided either by MACF or by application process at SSP.

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

4 Other signalling systems may be used.

FIGURE 5/Q.1218

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



NOTES

1 Transfer of correlation information needs to be supported.

2 Other signalling systems may be used.

FIGURE 6/Q.1218

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

Table 1 summarises the selection of features for each figure.

TABLE 1/Q.1218

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

Additional information related to each figure:

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

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

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

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

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

0.3 INAP protocol architecture

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

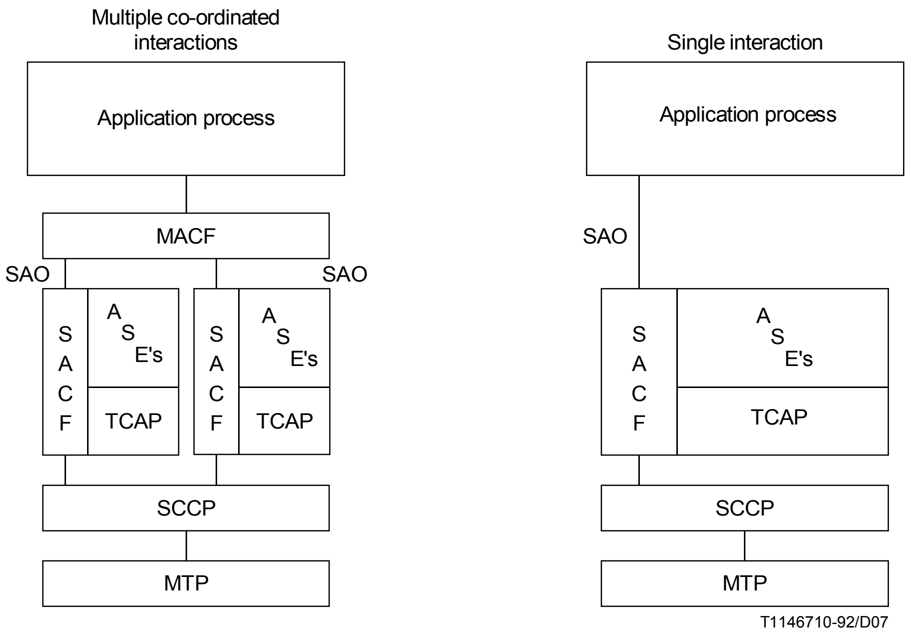
The INAP protocol architecture can be illustrated as shown in Figure 7.

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

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

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

Each ASE supports one or more operations. Description of each operation is tied with the action of corresponding FE modelling (see Recommendation Q.1214 and clause 3 of this Recommendation). Each operation is specified using the OPERATION macro described in Figure 8.



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

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

FIGURE 7/Q.1218

INAP protocol architecture

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

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

0.4 INAP addressing

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

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

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

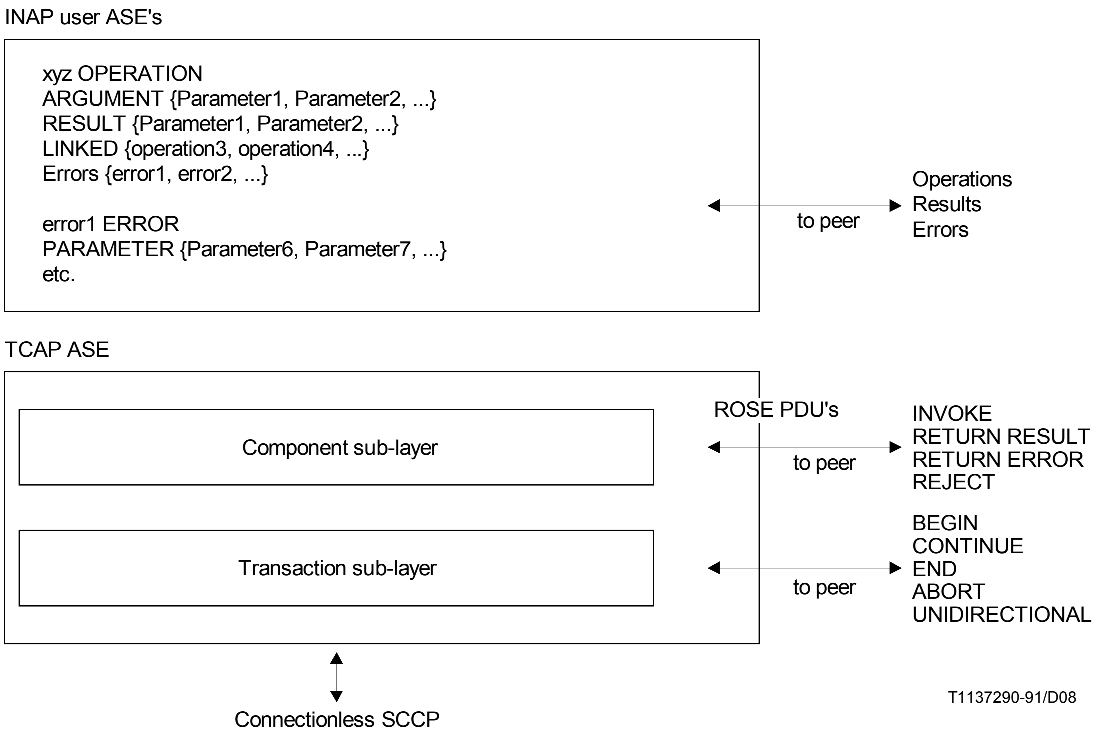


FIGURE 8/Q.1218

Operation description

0.5 Relationship between Recommendation Q.1214 and this Recommendation

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

Rec. Q.1214 Reference	*Information Flow*	*Operation*
6.4.2.1	Activate Service Filtering	Same
6.4.2.2	Activity Test	Same
6.4.2.3	Activity Test Response	Return Result from ActivityTest
6.4.2.4	Analysed Information	Same
6.4.2.5	Analyse Information	Same
6.4.2.6	Apply Charging	Same
6.4.2.7	Apply Charging Report	Same
6.4.2.8	Assist Request Instructions	Same
6.4.2.9	Call Gap	Same
6.4.2.10	Call Information Report	Same
6.4.2.11	Call Information Request	Same
6.4.2.12	Cancel Call Information Request	Cancel
6.4.2.13	Cancel Status Report Request	Same

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

Rec. Q.1214 Reference	*Information Flow*	*Operation*
6.4.2.49	TNoAnswer	Same
6.5.2.1	AssistRequestInstructions from SRF	AssistRequestInstructions
6.5.2.2	Cancel Announcement	Cancel
6.5.2.3	Collected User Information	Return Result from Prompt and collect user information
6.5.2.4	Play Announcement	Same
6.5.2.5	Prompt and collect user information	Same
6.5.2.6	Specialized Resource Report	Same
6.6.2.1	Query	Same
6.6.2.2	Query Result	Return Result from Query
6.6.2.3	SDF Response	Same
6.6.2.4	Update Confirmation	Return Result from Update Data
6.6.2.5	Update data	Same

1 SACF/MACF rules

1.1 Reflection of TCAP AC

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

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

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

1.2 Sequential/Parallel execution of operations

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

- charging operations may be synchronised with any other operation.

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

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

2 Abstract Syntax of the IN CS-1 Application Protocol

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

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

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

Class 1	Both RESULT and ERRORS appear in the ASN.1 OPERATION MACRO definition.
Class 2	Only ERRORS appears in the ASN.1 OPERATION MACRO definition.
Class 3	Only RESULT appears in the ASN.1 OPERATION MACRO definition.
Class 4	Neither RESULT nor ERRORS appears in the ASN.1 OPERATION MACRO definition.

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

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

The module containing all the type definitions for INAP operations is **IN-CS-1-Operations** and is described in 2.1.

The module containing all the type definitions for INAP errors is **IN-CS-1-Errors** and is described in 2.2.

The module containing all the type definitions for INAP data types is **IN-CS-1-DataTypes** and is described in 2.3.

The module containing the operation codes and error codes for INAP is **IN-CS-1-Codes** and is described in 2.4.

2.1 IN CS-1 Operation Types

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

        -- This module contains the type definitions for the IN CS-1 operations.
        -- There may be functional redundancies in the operation set related to call processing.
        -- This may make product interworking more difficult. Administrations wishing to deploy
        -- IN and equipment manufacturers implementing IN should take this into account.

DEFINITIONS ::=

BEGIN

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

-- error types
        Canceled,
        CancelFailed,
        DatabaseError,
        ETCFailed,
        ImproperCallerResponse,
        InfoKeyError,
        MissingCustomerRecord,
        MissingParameter,
        ParameterOutOfRange,
        Referral,
        RequestedInfoError,
        SystemFailure,
        TaskRefused,
        UnavailableResource,
        UnexpectedComponentSequence,
        UnexpectedDataValue,
        UnexpectedParameter,
        UnknownLegID,
        UnknownResource
```

```
FROM IN-CS-1-Errors { ccitt recommendation q 1218 modules(0) cs-1-errors(1) version1(0) }
-- argument types
            ActivateServiceFilteringArg,
            AnalysedInformationArg,
            AnalyseInformationArg,
            ApplyChargingArg,
            ApplyChargingReportArg,
            AssistRequestInstructionsArg,
            CallGapArg,
            CallInformationReportArg,
            CallInformationRequestArg,
            CancelArg,
            CancelStatusReportRequestArg,
            CollectedInformationArg,
            CollectInformationArg,
            ConnectArg,
            ConnectToResourceArg,
            EstablishTemporaryConnectionArg,
            EventNotificationChargingArg,
            EventReportBCSMArg,
            FurnishChargingInformationArg,
            HoldCallInNetworkArg,
            InitialDPArg,
            InitiateCallAttemptArg,
            MidCallArg,
            OAnswerArg,
            OCalledPartyBusyArg,
            ODisconnectArg,
            ONoAnswerArg,
            OriginationAttemptAuthorizedArg,
            PlayAnnouncementArg,
            PromptAndCollectUserInformationArg,
            QueryArg,
            QueryResultArg,
            ReceivedInformationArg,
            ReleaseCallArg,
            RequestCurrentStatusReportArg,
            RequestCurrentStatusReportResultArg,
            RequestEveryStatusChangeReportArg,
            RequestFirstStatusMatchReportArg,
            RequestNotificationChargingEventArg,
            RequestReportBCSMEventArg,
            ResetTimerArg,
            RouteSelectFailureArg,
            SelectFacilityArg,
            SelectRouteArg,
            SendChargingInformationArg,
            ServiceFilteringResponseArg,
            SpecializedResourceReportArg,
            StatusReportArg,
            TAnswerArg,
            TCalledPartyBusyArg,
            TDisconnectArg,
            TermAttemptAuthorizedArg,
            TNoAnswerArg,
            UpdateDataArg,
            UpdateDataResultArg
```

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

-- TYPE DEFINITIONS FOR   IN CS-1   OPERATIONS FOLLOWS

-- SCF-SSF operations

ActivateServiceFiltering ::= OPERATION
        ARGUMENT
            ActivateServiceFilteringArg
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedParameter
            }

-- SCF → SSF
-- When receiving this operation, the SSF handles calls to destination in a specified manner without
-- sending queries for every detected call. It is used for example for providing televoting or mass
-- calling services. Simple registration functionality (counters) and announcement control may be
-- located at the SSF. The operation initializes the specified counters in the SSF.

ActivityTest ::= OPERATION
        ARGUMENT
        RESULT

-- SCF → SSF
-- This operation is used to check for the continued existence of a relationship between the SCF
-- and SSF. If the relationship is still in existence, then the SSF will respond. If no reply is received,
-- then the SCF will assume that the SSF has failed in some way and will take the appropriate action.

AnalysedInformation ::= OPERATION
        ARGUMENT
            AnalysedInformationArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SSF → SCF
-- This operation is used to indicate availability of routing address and call type. (DP 3 – Analysed_Info).
-- For additional information on this operation and its use with open numbering plans, refer to 4.2.2.2a)3)/Q.1214.

AnalyseInformation ::= OPERATION
        ARGUMENT
            AnalyseInformationArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

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

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

-- SCF → SSF
-- This operation is used to interact with on line mechanisms that can be used for charging in the SSF.

ApplyChargingReport ::= OPERATION
        ARGUMENT
            ApplyChargingReportArg

-- SSF → SCF
-- This operation is used to report the results of charging in the SSF.

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

-- SSF → SCF or SRF → SCF
-- This operation is used when there is an assist or a hand-off procedure and may be sent by the SSF
-- or SRF to the SCF. This operation is sent by the SSF or SRF to the SCF, when the SSF has set up a
-- connection to the SRF as a result of receiving a ConnectToResource or EstablishTemporaryConnection
-- operation from the SCF. Refer to clause 3 for a description of the procedures associated with this operation.

CallGap ::= OPERATION
        ARGUMENT
            CallGapArg

-- SCF → SSF
-- This operation is used to request the SSF reduce the rate at which specific service requests are sent to
-- the SSF. Use of this operation by the SCF to gap queries and updates at the SDF is for further study.

CallInformationReport ::= OPERATION
        ARGUMENT
            CallInformationReportArg

-- SSF → SCF
-- This operation is used to send specific call information for a single call to the SCF as requested by the SCF
-- in a previous callInformationRequest.

CallInformationRequest ::= OPERATION
        ARGUMENT
            CallInformationRequestArg
        ERRORS {
            Canceled,
            MissingParameter,
            ParameterOutOfRange,
            RequestedInfoError,
            SystemFailure,
            TaskRefused,
                    UnexpectedComponentSequence,
                    UnexpectedParameter
                    }
-- SCF → SSF
-- This operation is used to request the SSF to save specific information about a single call and report it to
-- the SCF at the end of the call (with a callInformationReport operation).

Cancel ::= OPERATION
              ARGUMENT
                    CancelArg
              ERRORS {
                    CancelFailed
                    }
-- SCF → SSF, or SCF → SRF
-- This generic operation cancels the correlated previous operation. The following operations can be canceled:
-- PlayAnnouncement, PromptAndCollectUserInformation, and CallInformationRequest.

CancelStatusReportRequest ::= OPERATION
              ARGUMENT
                    CancelStatusReportRequestArg
              ERRORS {
                    CancelFailed
                    }
-- SCF → SSF
-- This operation cancels the following processes: RequestFirstStatusMatchReport and
-- RequestEveryStatusChangeReport.

CollectedInformation ::= OPERATION
              ARGUMENT
                    CollectedInformationArg
              ERRORS {
                    MissingCustomerRecord,
                    MissingParameter,
                    SystemFailure,
                    TaskRefused,
                    UnexpectedComponentSequence,
                    UnexpectedDataValue,
                    UnexpectedParameter
                    }
-- SSF → SCF
-- This operation is used to indicate availability of complete initial information package/dialing string from
-- originating party. (This event may have already occurred in the case of en bloc signaling, in which case
-- the waiting duration in this PIC is zero.) (DP 2 – Collected_Info). For additional information on this operation
-- and its use with open numbering plans, refer to 4.2.2.2a)2)/Q.1214.

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

```
Connect ::= OPERATION
        ARGUMENT
            ConnectArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SCF → SSF
-- This operation is used to request the SSF to perform the call processing actions to route or forward a call to
-- a specified destination. To do so, the SSF may or may not use destination information from the calling party
-- (e.g. dialed digits) and existing call setup information (e.g. route index to a list of trunk groups), depending on
-- the information provided by the SCF.

ConnectToResource ::= OPERATION
        ARGUMENT
            ConnectToResourceArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SCF → SSF
-- This operation is used to connect a call from the SSP to the physical entity containing the SRF.
-- Refer to clause 3 for a description of the procedures associated with this operation.

Continue ::= OPERATION
        ARGUMENT

-- SCF → SSF
-- This operation is used to request the SSF to proceed with call processing at the DP at which it
-- previously suspended call processing to await SCF instructions (i.e. proceed to the next point
-- in call in the BCSM). The SSF continues call processing without substituting new data from SCF.

DisconnectForwardConnection ::= OPERATION
        ARGUMENT
        ERRORS {
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedParameter
            }

-- SCF → SSF
-- This operation is used to disconnect a forward temporary connection and a connection to a resource.
-- Refer to clause 3 for a description of the procedures associated with this operation.

EstablishTemporaryConnection ::= OPERATION
        ARGUMENT
            EstablishTemporaryConnectionArg
        ERRORS {
            ETCFailed,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

```
-- SCF → SSF
-- This operation is used to create a connection to a resource for a limited period of time
-- (e.g. to play an announcement, to collect user information); it implies the use of the assist
-- procedure. Refer to clause 3 for a description of the procedures associated with this operation.

EventNotificationCharging ::= OPERATION
        ARGUMENT
            EventNotificationChargingArg

-- SSF → SCF
-- This operation is used to notify the SCF of a charging-related event previously requested
-- by the SCF in a RequestNotificationChargingEvent operation.

EventReportBCSM ::= OPERATION
        ARGUMENT
            EventReportBCSMArg

-- SSF → SCF
-- This operation is used to notify the SCF of a call-related event (e.g. BCSM events such as busy or
-- no answer) previously requested by the SCF in a RequestReportBCSMEvent operation.

FurnishChargingInformation ::= OPERATION
        ARGUMENT
            FurnishChargingInformationArg
        ERRORS {
            MissingParameter,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SCF → SSF
-- This operation is used to give some charging information to the SSF, to be used later in off line processing.

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

-- SCF → SSF
-- This operation is used to provide the capability of queueing a call during the setup phase (e.g. to provide
-- a call completion to busy, the call would be queued until the destination becomes free).

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

-- SSF → SCF
-- This operation is used after a TDP to indicate request for service.
```

```
InitiateCallAttempt ::= OPERATION
        ARGUMENT
            InitiateCallAttemptArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to request the SSF to create a new call to one or more call parties using address information
-- provided by the SCF (e.g. wake-up call, predefined conference call, previous prompt and collect information).

OAnswer ::= OPERATION
        ARGUMENT
            OAnswerArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for indication from the terminating half BCSM that the call is accepted and answered
-- by terminating party (e.g. terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer
-- message received) (DP 7 – O_Answer). For additional information on this operation, refer to 4.2.2.2a)4)/Q.1214.

OCalledPartyBusy ::= OPERATION
        ARGUMENT
            OCalledPartyBusyArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for Indication from the terminating half BCSM that the terminating party is busy
-- (DP 5 – O_Called_Party_Busy). For additional information on this operation, refer to 4.2.2.2a)4)/Q.1214.

ODisconnect ::= OPERATION
        ARGUMENT
            ODisconnectArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message, SS7 Release message)
-- is received from the originating party, or received from the terminating party via the terminating half BCSM.
-- (DP 9 – O_Disconnect). For additional information on this operation, refer to 4.2.2.2a)5)/Q.1214.
```

```
OMidCall ::= OPERATION
        ARGUMENT
            MidCallArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used to indicate a feature request is received from the originating party
-- (e.g. hook flash, ISDN feature activator, Q.931 HOLD or RETrieve message). (DP 8 – O_Mid_Call).
-- For additional information on this operation, refer to 4.2.2.2a)5)/Q.1214.

ONoAnswer ::= OPERATION
        ARGUMENT
            ONoAnswerArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for indication from the terminating half BCSM that the terminating party does not
-- answer within a specified time period (DP 6 – O_No_Answer). For additional information on this operation,
-- refer to 4.2.2.2a)4)/Q.1214.

OriginationAttemptAuthorized ::= OPERATION
        ARGUMENT
            OriginationAttemptAuthorizedArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used to Indicate the desire to place outgoing call (e.g. offhook, Q.931 Setup message,
-- ISDN-UP IAM message) and authority/ability to place outgoing call verified (DP 1 – Origination_Attempt_Authorized).
-- For additional information on this operation, refer to 4.2.2.2a)1)/Q.1214.

ReleaseCall ::= OPERATION
        ARGUMENT
            ReleaseCallArg
-- SCF → SSF
-- This operation is used to tear down an existing call at any phase of the call for 1, 2, or more parties
-- involved in the call.

RequestCurrentStatusReport ::= OPERATION
        ARGUMENT
            RequestCurrentStatusReportArg
        RESULT
            RequestCurrentStatusReportResultArg
```

```
          ERRORS {
               MissingParameter,
               ParameterOutOfRange,
               SystemFailure,
               TaskRefused,
               UnexpectedComponentSequence,
               UnexpectedParameter,
               UnknownResource
               }
-- SCF → SSF
-- This operation is used to request the SSF to report immediately the busy/idle status of a physical
-- termination resource.

RequestEveryStatusChangeReport ::= OPERATION
          ARGUMENT
               RequestEveryStatusChangeReportArg
          RESULT
          ERRORS {
               Canceled,
               MissingParameter,
               ParameterOutOfRange,
               SystemFailure,
               TaskRefused,
               UnexpectedComponentSequence,
               UnexpectedParameter,
               UnknownResource
               }
-- SCF → SSF
-- This operation is used to request the SSF to report every change of busy/idle status of a physical
-- termination resource.

RequestFirstStatusMatchReport ::= OPERATION
          ARGUMENT
               RequestFirstStatusMatchReportArg
          RESULT
          ERRORS {
               Canceled,
               MissingParameter,
               ParameterOutOfRange,
               SystemFailure,
               TaskRefused,
               UnexpectedComponentSequence,
               UnexpectedParameter,
               UnknownResource
               }
-- SCF → SSF
-- This operation is used to request the SSF to report the first change busy/idle to the specified status of
-- a physical termination resource.

RequestNotificationChargingEvent ::= OPERATION
          ARGUMENT
               RequestNotificationChargingEventArg
          ERRORS {
               MissingParameter,
               SystemFailure,
               TaskRefused,
               UnexpectedComponentSequence,
               UnexpectedDataValue,
               UnexpectedParameter
               }
-- SCF → SSF
-- This operation is used to request the SSF to monitor for a charging-related event, then send a notification back
-- to the SCF when the event is detected.
```

```
RequestReportBCSMEvent ::= OPERATION
        ARGUMENT
            RequestReportBCSMEventArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to request the SSF to monitor for a call-related event (e.g. BCSM events such as
-- busy or no answer), then send a notification back to the SCF when the event is detected.

ResetTimer ::= OPERATION
        ARGUMENT
            ResetTimerArg
        ERRORS {
            MissingParameter,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to request the SSF to refresh an application timer in the SSF.

RouteSelectFailure ::= OPERATION
        ARGUMENT
            RouteSelectFailureArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used to indicate that the SSP is unable to select a route (e.g. unable to determine a
-- correct route, no more routes on route list) or indication from the terminating half BCSM that a call
-- cannot be presented to the terminating party (e.g. network ongestion) (DP 4 – Route_Select_Failure).
-- For additional information on this operation, refer to 4.2.2.2a)4)/Q.1214.

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

```
SelectRoute ::= OPERATION
        ARGUMENT
            SelectRouteArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to request the SSF to perform the originating basic call processing actions to
-- determine routing information and select a route for a call, based either on call information available
-- to the SSF, or on call information provided by the SCF (e.g. for alternate routing), to include the
-- called party address, type of call, carrier, route index, and one or more alternate route indices.
-- Based on the routing information, the SSF attempts to select a primary route for the call, and if the
-- route is busy, attempts to select an alternate route. The SSF may fail to select a route for the call
-- if all routes are busy.

SendChargingInformation ::= OPERATION
        ARGUMENT
            SendChargingInformationArg
        ERRORS {
            MissingParameter,
            UnexpectedComponentSequence,
            UnexpectedParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnknownLegID
            }
-- SCF → SSF
-- This operation is used to request the SSF to generate specific charging messages.

ServiceFilteringResponse ::= OPERATION
        ARGUMENT
            ServiceFilteringResponseArg
-- SSF → SCF
-- This operation is used to send back to the SCF the values of counters specified in a previous
-- ActivateServiceFiltering operation.

StatusReport ::= OPERATION
        ARGUMENT
            StatusReportArg
-- SSF → SCF
-- This operation is used as a response to RequestFirstStatusMatchReport or
-- RequestEveryStatusChangeReport operations.

TAnswer ::= OPERATION
        ARGUMENT
            TAnswerArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used to indicate that the call is accepted and answered by terminating party
-- (e.g. terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer message
-- received) (DP 15 – T_Answer). For additional information on this operation, refer to 4.2.2.2b)8)/Q.1214.
```

```
TCalledPartyBusy ::= OPERATION
        ARGUMENT
            TCalledPartyBusyArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used to indicate all resources in group busy (DP 13 – TCalledPartyBusy).
-- For additional information on this operation, refer to 4.2.2.2b)8)/Q.1214.

TDisconnect ::= OPERATION
        ARGUMENT
            TDisconnectArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message,
-- SS7 Release message) is received from the terminating party, or received from the originating party
-- via the originating half BCSM. (DP 17 – T_Disconnect). For additional information on this operation,
-- refer to 4.2.2.2b)10)/Q.1214.

TermAttemptAuthorized ::= OPERATION
        ARGUMENT
            TermAttemptAuthorizedArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SSF → SCF
-- This operation is used for indication of incoming call received from originating half BCSM and authority
-- to route call to a specified terminating resource (or group) verified. (DP 12 – Termination_Authorized).
-- For additional information on this operation, refer to 4.2.2.2b)7)/Q.1214.

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

```
-- SSF → SCF
-- This operation is used to indicate that a feature request is received from the terminating party (e.g. hook
-- flash, ISDN feature activator, Q.931 HOLD or RETrieve message). (DP 16 – T_Mid_Call).
-- For additional information on this operation, refer to 4.2.2.2a)10)/Q.1214.

TNoAnswer ::= OPERATION
        ARGUMENT
            TNoAnswerArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SSF → SCF
-- This operation is used to indicate that the terminating party does not answer within a specified duration.
-- (DP 14 – T_No_Answer). For additional information on this operation, refer to 4.2.2.2b)9)/Q.1214.

-- SCF-SRF operations

-- AssistRequestInstructions

-- SRF → SCF
-- Refer to previous description of this operation in the SCF-SSF operations section Cancel

-- SCF → SRF
-- Refer to previous description of this operation in the SCF-SSF operations section.

PlayAnnouncement ::= OPERATION
        ARGUMENT
            PlayAnnouncementArg
        ERRORS {
            Canceled,
            MissingParameter,
            SystemFailure,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnavailableResource
            }
        LINKED {
            SpecializedResourceReport
            }

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

PromptAndCollectUserInformation ::= OPERATION
        ARGUMENT
            PromptAndCollectUserInformationArg
        RESULT
            ReceivedInformationArg
        ERRORS {
            Canceled,
            ImproperCallerResponse,
            MissingParameter,
            SystemFailure,
            TaskRefused,
                UnavailableResource,
                UnexpectedDataValue
                }
-- SCF → SRF
-- This operation is used to interact with a user to collect information.

SpecializedResourceReport ::= OPERATION
        ARGUMENT
                SpecializedResourceReportArg

-- SRF → SCF
-- This operation is used as the response to a PromptAndCollectUserInformation operation or as the response to
-- a PlayAnnouncement operation when the announcement completed report indication is set.
-- SCF-SDF operations

Query ::= OPERATION
        ARGUMENT
                QueryArg
        RESULT
                QueryResultArg
        ERRORS {
                DatabaseError,
                InfoKeyError,
                MissingParameter,
                Referral,
                RequestedInfoError,
                UnexpectedDataValue,
                UnexpectedParameter
                }
        LINKED {
                SdfResponse
                }
-- SCF → SDF
-- This operation is used to query an item of data held in the SDF (e.g. a translation of a Freephone number).
-- The timer associated with this operation must be of a sufficient duration to allow its linked operation
-- to be correctly correlated.

SdfResponse ::= OPERATION
        ARGUMENT

-- SDF → SCF
-- This operation is used to acknowledge the start of a Query or Update operation.

UpdateData ::= OPERATION
        ARGUMENT
                UpdateDataArg
        RESULT
                UpdateDataResultArg
        ERRORS {
                DatabaseError,
                InfoKeyError,
                MissingParameter,
                Referral,
                RequestedInfoError,
                UnexpectedDataValue,
                UnexpectedParameter
                }
        LINKED {
                SdfResponse
                }
-- SCF → SDF
-- This operation is used to update an item of data held in the SDF.

END
```

2.2 IN CS-1 Error Types

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

-- This module contains the type definitions for the IN CS-1 errors.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
        ERROR

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

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

-- TYPE DEFINITION FOR    IN CS-1    ERRORS FOLLOWS

Canceled ::= ERROR

-- The operation has been canceled.

CancelFailed ::= ERROR
        PARAMETER SEQUENCE {
            problem     [0] ENUMERATED {
                                unknownOperation(0),
                                tooLate(1),
                                operationNotCancellable(2)
                                },
            operation   [1] InvokeID
        }

-- The operation failed to be canceled.

DatabaseError ::= ERROR
        PARAMETER ENUMERATED {
            invalidDatabaseID(1),
            databaseCurrentlyUnavailable(2),
            databaseDenied(3),
            databaseRequestDenied(4)
            -- other values FFS
            }

-- The SDF database could not be accessed.

ETCFailed ::= ERROR

-- The establish temporary connection failed.

ImproperCallerResponse ::= ERROR

-- The caller response was not as expected.

InfoKeyError ::= ERROR
        PARAMETER SEQUENCE {
            errorType   [0] ENUMERATED {
                                invalidInfoKey(1),
                                infoKeyNotUnique(2)
                                -- other values FFS
                                },
            attributeID [1] AttributeID }

-- The information key is invalid or ambiguous.

MissingCustomerRecord ::= ERROR

-- The Service Logic Program could not be found in the SCF.
```

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

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

Referral ::= ERROR      -- reference to other database.
          PARAMETER SEQUENCE {
               databaseAddress    [0] OCTET STRING,
               databaseID[1] DatabaseID          OPTIONAL
               }
-- The SDF accessed does not have the data requested and instead refers the requesting
-- entity to another SDF.

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

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

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

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

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

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

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

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

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

END
```

2.3 IN CS-1 data types

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

-- This module contains the type definitions for the IN CS-1 data types.

-- The following parameters map onto bearer protocol (i.e. Q.931, case 2 and ISUP) parameters:
-- CallingPartyBusinessGroupID, CallingPartySubaddress, CalledPartyNumber,
-- CalledPartyLineID, Prefix (derived from dialed digits), DestinationRoutingAddress,
-- DialedDigits, AccessTransport, CallingPartyCategory, LocationNumber,
-- TravellingClassMark, AssistingSSPIPRoutingAddress, AlertingPattern (Q.931 only),
-- ReleaseCause (and other Cause parameters), and SPID (Q.931 only).

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

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

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
        ExtensionField
FROM { ccitt recommendation q 1400 modules(0) extension-example(1) version1(0) };

-- TYPE DEFINITIONS FOR   IN CS-1   DATA TYPES FOLLOWS

-- Argument data types

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

ActivateServiceFilteringArg ::= SEQUENCE {
      filteredCallTreatment             [0]  FilteredCallTreatment,
      filteringCharacteristics          [1]  FilteringCharacteristics,
      filteringTimeOut                  [2]  FilteringTimeOut,
      filteringCriteria                 [3]  FilteringCriteria            OPTIONAL,
      startTime                         [4]  DateAndTime                  OPTIONAL,
      extensions                        [5]  SEQUENCE SIZE(0..MAX)        OF ExtensionField OPTIONAL
      }
-- filteringCriteria is required if this operation is not in the context of a call as determined by the SCF.
-- Depending on implementation choices, the filteringCriteria parameter can be used to correlate the
-- ServiceFilteringResponse with ActivateServiceFiltering. In this case, filteringCriteria parameter
-- is always required.

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

```
AnalyseInformationArg ::= SEQUENCE {
        destinationRoutingAddress         [0]  DestinationRoutingAddress,
        alertingPattern                   [1]  AlertingPattern                OPTIONAL,
        iSDNAccessRelatedInformation      [2]  ISDNAccessRelatedInformation   OPTIONAL,
        originalCalledPartyID             [3]  OriginalCalledPartyID          OPTIONAL,
        extensions                        [4]  SEQUENCE SIZE(0..MAX)          OF ExtensionField OPTIONAL
        }

ApplyChargingArg ::= SEQUENCE {
        billingChargingCharacteristics    [0]  BillingChargingCharacteristics,
        sendCalculationToSCFIndication    [1]  BOOLEAN                        DEFAULT FALSE,
        partyToCharge                     [2]  LegID                          OPTIONAL,
        extensions                        [3]  SEQUENCE SIZE(0..MAX)          OF ExtensionField OPTIONAL
        }

ApplyChargingReportArg ::= CallResult

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

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

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

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

CancelArg ::= invokeID [0] InvokeID
-- The InvokeID has the same value as that which was used for the operation to be cancelled.

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

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

CollectInformationArg ::= SEQUENCE {
        alertingPattern                 [0]  AlertingPattern               OPTIONAL,
        numberingPlan                   [1]  NumberingPlan                 OPTIONAL,
        originalCalledPartyID           [2]  OriginalCalledPartyID         OPTIONAL,
        travellingClassMark             [3]  TravellingClassMark           OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(0..MAX)         OF ExtensionField OPTIONAL
        }

ConnectArg ::= SEQUENCE {
        destinationRoutingAddress       [0]  DestinationRoutingAddress,
        alertingPattern                 [1]  AlertingPattern               OPTIONAL,
        correlationID                   [2]  CorrelationID                 OPTIONAL,
        cutAndPaste                     [3]  CutAndPaste                   OPTIONAL,
        forwardingCondition             [4]  ForwardingCondition           OPTIONAL,
        iSDNAccessRelatedInformation    [5]  ISDNAccessRelatedInformation  OPTIONAL,
        originalCalledPartyID           [6]  OriginalCalledPartyID         OPTIONAL,
        routeList                       [7]  RouteList                     OPTIONAL,
        scfID                           [8]  ScfID                         OPTIONAL,
        travellingClassMark             [9]  TravellingClassMark           OPTIONAL,
        extensions                      [10] SEQUENCE SIZE(0..MAX)         OF ExtensionField OPTIONAL
        }
-- For alerting pattern, OPTIONAL denotes that this parameter only applies if SSF is the terminating local
-- exchange for the subscriber.

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

DpSpecificCommonParameters ::= SEQUENCE {
        serviceAddressInformation       [0]  ServiceAddressInformation,
        bearerCapability                [1]  BearerCapability              OPTIONAL,
        calledPartyNumber               [2]  CalledPartyNumber             OPTIONAL,
        callingLineID                   [3]  CallingPartyNumber            OPTIONAL,
        callingPartysCategory           [4]  CallingPartysCategory         OPTIONAL,
        iPSSPCapabilities               [5]  IPSSPCapabilities             OPTIONAL,
        iPAvailable                     [6]  IPAvailable                   OPTIONAL,
        iSDNAccessRelatedInformation    [7]  ISDNAccessRelatedInformation  OPTIONAL,
        cGEncountered                   [8]  CGEncountered                 OPTIONAL,
        locationNumber                  [9]  LocationNumber                OPTIONAL,
        serviceProfileIdentifier        [10] ServiceProfileIdentifier      OPTIONAL,
        terminalType                    [11] TerminalType                  OPTIONAL,
        extensions                      [12] SEQUENCE SIZE(0..MAX)         OF ExtensionField OPTIONAL
        }
-- OPTIONAL for iPSSPCapabilities, iPAvailable, and cGEncountered denotes network operator specific use.
-- OPTIONAL for dialledDigits, callingLineID, and callingLineCategory refer to clause 3 for the trigger detection
-- point processing rules to specify when these parameters are included in the message.BearerCapability
-- should be appropriately coded as speech.
```

```
EstablishTemporaryConnectionArg ::= SEQUENCE {
        assistingSSPIPRoutingAddress     [0]  AssistingSSPIPRoutingAddress,
        correlationID                    [1]  CorrelationID                    OPTIONAL,
        legID                            [2]  LegID                            OPTIONAL,
        scfID                            [3]  ScfID                            OPTIONAL,
        extensions                       [4]  SEQUENCE SIZE(0..MAX)            OF ExtensionField OPTIONAL
        }

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

EventReportBCSMArg ::= SEQUENCE {
        eventTypeBCSM                    [0]  EventTypeBCSM,
        bcsmEventCorrelationID           [1]  CorrelationID                    OPTIONAL,
        eventSpecificInformationBCSM     [2]  EventSpecificInformationBCSM     OPTIONAL,
        legID                            [3]  LegID                            OPTIONAL,
        miscCallInfo                     [4]  MiscCallInfo                     OPTIONAL,
        extensions                       [5]  SEQUENCE SIZE(0..MAX)            OF ExtensionField OPTIONAL
        }
-- OPTIONAL denotes network operator specific use.

FurnishChargingInformationArg ::= BillingChargingCharacteristics

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

InitialDPArg ::= SEQUENCE {
        serviceKey                       [0]  ServiceKey,
        dialledDigits                    [1]  CalledPartyNumber                OPTIONAL,
        calledPartyNumber                [2]  CalledPartyNumber                OPTIONAL,
        callingLineID                    [3]  CallingPartyNumber               OPTIONAL,
        callingPartyBusinessGroupID      [4]  CallingPartyBusinessGroupID      OPTIONAL,
        callingPartysCategory            [5]  CallingPartysCategory            OPTIONAL,
        callingPartySubaddress           [6]  CallingPartySubaddress           OPTIONAL,
        cGEncountered                    [7]  CGEncountered                    OPTIONAL,
        iPSSPCapabilities                [8]  IPSSPCapabilities                OPTIONAL,
        iPAvailable                      [9]  IPAvailable                      OPTIONAL,
        locationNumber                   [10] LocationNumber                   OPTIONAL,
        miscCallInfo                     [11] MiscCallInfo                     OPTIONAL,
        originalCalledPartyID            [12] OriginalCalledPartyID            OPTIONAL,
        serviceProfileIdentifier         [13] ServiceProfileIdentifier         OPTIONAL,
        terminalType                     [14] TerminalType                     OPTIONAL,
        extensions                       [15] SEQUENCE SIZE(0..MAX)            OF ExtensionField OPTIONAL
        }
-- OPTIONAL for iPSSPCapabilities, iPAvailable, cGEncountered, and miscCallInfo denotes network
-- operator specific use.
-- OPTIONAL for dialledDigits, callingLineID, and callingLineCategory refer to clause 3 for the trigger
-- detection point processing rules to specify when these parameters are included in the message.
-- OPTIONAL for terminalType indicates that this parameter applies only at originating or terminating
-- local exchanges if the SSF has this information.

InitiateCallAttemptArg ::= SEQUENCE {
        destinationRoutingAddress        [0]  DestinationRoutingAddress,
        alertingPattern                  [1]  AlertingPattern                  OPTIONAL,
        iSDNAccessRelatedInformation     [2]  ISDNAccessRelatedInformation     OPTIONAL,
        travellingClassMark              [3]  TravellingClassMark              OPTIONAL,
        extensions                       [4]  SEQUENCE SIZE(0..MAX)            OF ExtensionField OPTIONAL
        }
```

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

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

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

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

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

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

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

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

QueryArg ::= SEQUENCE {
        informationKey                 [0]  SET OF Attribute,
        databaseID                     [1]  DatabaseID                     OPTIONAL,
        requestedInfoType              [2]  SET OF AttributeID             OPTIONAL,
        extensions                     [3]  SEQUENCE SIZE(0..MAX)          OF ExtensionField OPTIONAL
        }
-- requestedInfoType is used when the requested info type is known by the SDF.

QueryResultArg ::= SET OF Attribute

ReceivedInformationArg ::= CHOICE {
        digitsResponse                 [0]  OCTET STRING,
        iA5Response                    [1]  IA5String
        }

ReleaseCallArg ::= Cause

-- A default value of decimal 31 (normal unspecified) should be coded appropriately.

RequestCurrentStatusReportArg ::= ResourceID

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

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

RequestFirstStatusMatchReportArg ::= SEQUENCE {
        resourceID                     [0]  ResourceID,
        resourceStatus                 [1]  ResourceStatus,
        correlationID                  [2]  CorrelationID               OPTIONAL,
        monitorDuration                [3]  Duration                    OPTIONAL,
        extensions                     [4]  SEQUENCE SIZE(0..MAX)       OF ExtensionField OPTIONAL
        }
-- For correlationID OPTIONAL denotes network operator optional.
-- monitorDuration is required if outside the context of a call. It is not expected if we are in the context
-- of a call, because in that case the end of the call implicitly means the end of the monitoring.

RequestNotificationChargingEventArg ::= SEQUENCE SIZE(0..MAX)        OF ChargingEvent

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

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

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

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

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

SendChargingInformationArg ::= SEQUENCE {
        billingChargingCharacteristics  [0]  BillingChargingCharacteristics,
        legID                           [1]  LegID,
        extensions                      [2]  SEQUENCE SIZE(0..MAX)        OF ExtensionField OPTIONAL
        }

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

SpecializedResourceReportArg ::=   NULL

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

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

TCalledPartyBusyArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        busyCause                       [1]  Cause                        OPTIONAL,
        calledPartyBusinessGroupID      [2]  CalledPartyBusinessGroupID   OPTIONAL,
        calledPartySubaddress           [3]  CalledPartySubaddress        OPTIONAL,
        originalCalledPartyID           [4]  OriginalCalledPartyID        OPTIONAL,
        redirectingPartyID              [5]  RedirectingPartyID           OPTIONAL,
        redirectionInformation          [6]  RedirectionInformation       OPTIONAL,
        routeList                       [7]  RouteList                    OPTIONAL,
        travellingClassMark             [8]  TravellingClassMark          OPTIONAL,
        extensions                      [9]  SEQUENCE SIZE(0..MAX)        OF ExtensionField OPTIONAL
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.
```

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

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

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

UpdateDataArg  ::= SEQUENCE {
        informationKey                  [0]  SET OF Attribute,
        updatedInfo                     [1]  SET OF Attribute,
        databaseID                      [2]  DatabaseID                     OPTIONAL,
        extensions                      [3]  SEQUENCE SIZE(0..MAX)          OF ExtensionField OPTIONAL
        }
```

-- The function type is coded in the informationKey.

UpdateDataResultArg ::= SEQUENCE SIZE (0..MAX)   OF Attribute

-- The ***Definition of Common Data Types*** *Follows*

AlertingPattern ::= OCTET STRING

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

ApplicationTimer ::= INTEGER

-- Used by the SCF to set a timer in the SSF. The timer is in seconds.

AssistingSSPIPRoutingAddress ::= Digits

-- Indicates the destination address of the SRF for the assist procedure.

```
Attribute ::= SEQUENCE {
        attributeID                     [0]  AttributeID,
        attributeValues                 [1]  AttributeValue
        }
```

AttributeID ::= INTEGER

```
AttributeValue ::= ANY DEFINED BY attributeID

-- currently defined attribute types:
--                                 AttributeID      Type of Attribute value
-- freeAttribute                        0           OCTET STRING (SIZE(0..MAX))
-- functionType                         1           ENUMERATED (
--                                                          update(0),
--                                                          increment(1),
--                                                          decrement(2))
-- calledAddress                        2           Digits
-- callingAddress                       3           Digits
-- personalIDNumber                     4           Digits
-- accountNumber                        5           Digits
-- origAdministrationID                 6           Digits
-- bearerCapability                     7           BearerCapability
-- callingTermCapabilities              8           TermCapability
-- onNetOffNetIndicator                 9           ENUMERATED {
--                                                          on(0),
--                                                          off(1),
--                                                          undetermined(2) }
-- other types FFS

BCSMEvent ::= SEQUENCE {
        eventTypeBCSM                   [0]  EventTypeBCSM,
        monitorMode                     [1]  MonitorMode,
        legID                           [2]  LegID                          OPTIONAL,
        applicationTimer                [3]  ApplicationTimer               OPTIONAL
        }

-- Indicates the BCSM Event information for monitoring. The SCF may set a timer in the SSF for the
-- No Answer event. If the user doesn't answer the call within the allotted time, the SSF reports the event
-- to the SCF.

BearerCapability ::= CHOICE {
        bearerCapability                [0]  OCTET STRING (SIZE(2..MAX)),
        tmr                             [1]  OCTET STRING (SIZE(1))
        }

-- Indicates the type of bearer capability connection to the user. For bearerCapability, either
-- DSS 1 (Q.931) or the ISUP User Service Information (Q.763) encoding can be used. Refer
-- to the Q.763 Transmission Medium Requirement parameter for tmr encoding.

BillingChargingCharacteristics ::= OCTET STRING

-- Indicates the billing and/or charging calculation characteristics. The value of this octet string is
-- network operator specific.

CalledPartyBusinessGroupID ::= OCTET STRING

-- Indicates the business group of the called party. The value of this octet string is network
-- operator specific.

CalledPartyNumber ::= OCTET STRING

-- Indicates the Called Party Number. Refer to Q.763 for encoding.

CalledPartySubaddress ::= OCTET STRING

-- Indicates the Called Party Subaddress. Refer to Q.931 for encoding.

CallingPartyBusinessGroupID ::= OCTET STRING

-- Indicates the business group of the calling party. The value of this octet string is network
-- operator specific.

CallingPartyNumber ::= OCTET STRING

-- Indicates the Calling Party Number. Refer to Q.763 for encoding.

CallingPartySubaddress ::= OCTET STRING

-- Indicates the Calling Party Subaddress. Refer to Q.931 for encoding.

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

-- Indicates the type of calling party (e.g. operator, payphone, ordinary subscriber). Refer to Q.763
-- for encoding.
```

```
CallResult ::= OCTET STRING

-- Indicates call result for billing purposes. The value of this octet string is network operator specific.

Cause ::= OCTET STRING

-- Indicates the cause for interface related information. Refer to the Q.763 Cause parameter for encoding.

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

-- Indicates the type of automatic code gapping encountered, if any.

ChargingEvent ::= SEQUENCE {
        eventTypeCharging               [0]  EventTypeCharging,
        monitorMode                     [1]  MonitorMode,
        legID                           [2]  LegID                      OPTIONAL
        }

-- Indicates the charging event information for monitoring.

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

-- The use of voiceBack is network operator specific.
-- The endOfReplyDigit, cancelDigit, and startDigit parameters have been designated as OCTET STRING,
-- and are to be encoded as Q.763 address signal digits in the Called Party Number. The endOfReplyDigit,
-- cancelDigit, and startDigit are one digit only with: code 11 = *, code 12 = #.

CollectedInfo ::= CHOICE {
        collectedDigits                 [0]  CollectedDigits,
        iA5Information                  [1]  BOOLEAN (TRUE)
        }

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

CorrelationID ::= Digits

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

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

CounterID ::= INTEGER (0..99)

-- Indicates the counters to be incremented.
-- The counterIDs can be addressed by using the last digits of the dialed number.

CountersValue ::= SEQUENCE SIZE(0..99) OF CounterAndValue

CutAndPaste ::= INTEGER

-- Indicates the number of digits to be deleted. Refer to 6.4.2.16/Q.1214 for additional information.
```

DatabaseID ::= OCTET STRING

-- Indicates the database to query. The value of this octet string is network operator specific.

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

-- Indicates the start time for activate service filtering. Coded as YYMMDDHHMMSS with each digit coded BCD.
-- The first octet contains YY and the remaining items are sequenced following.

DestinationRoutingAddress ::= SEQUENCE SIZE(1..3) OF CalledPartyNumber

-- Indicates the list of Called Party Numbers (primary and alternates).

Digits ::= OCTET STRING

-- Indicates the address signalling digits. Refer to the Q.763 Generic Number and Generic Digits parameters
-- for encoding. The following parameter should use Generic Number:
-- CorrelationID for AssistRequestInstructions. The following parameters should use Generic Digits: prefix, all
-- other CorrelationID occurrences, personalIDNumber attribute, accountNumber attribute, origAdministrationID
-- attribute, calledAddress attribute, callingAddress attribute, dialledNumber filtering criteria, callingLineID
-- filtering criteria, lineID ResourceID type.

DisplayInformation ::= IA5String

-- Indicates the display information.

Duration ::= INTEGER (0..2047)

-- Values are seconds

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

EventSpecificInformationBCSM ::= OCTET STRING *-- defined by network operator*.

-- Indicates the call related information specific to the event.

EventSpecificInformationCharging ::= OCTET STRING *-- defined by network operator*.

-- Indicates the charging related information specific to the event.

```
EventTypeBCSM ::= ENUMERATED {
        origAttemptAuthorized(1),
        collectInfo(2),
        analysedInformation(3),
        routeSelectFailure(4),
        oCalledPartyBusy(5),
        oNoAnswer(6),
        oAnswer(7),
        oMidCall(8),
        oDisconnect(9),
        oAbandon(10),
        reserved(11),
        termAttemptAuthorized(12),
        tCalledPartyBusy(13),
        tNoAnswer(14),
        tAnswer(15),
        tMidCall(16),
        tDisconnect(17),
        tAbandon(18)
        }
```

-- Indicates the BCSM detection point event. Refer to 4.2.2.2/Q.1214 for additional information on the events.

EventTypeCharging ::= OCTET STRING

-- Indicates the charging event. Network operator specific based on Q.763 Charging Indicators message.

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

FacilityGroupMember ::= INTEGER

-- Indicates the specific member of a trunk group or multi-line hunt group.

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

FilteredCallTreatment ::= SEQUENCE {
        billingChargingCharacteristics  [0]  BillingChargingCharacteristics,
        informationToSend               [1]  InformationToSend              OPTIONAL,
        maximumNumberOfCounters         [2]  MaximumNumberOfCounters        OPTIONAL,
        releaseCause                    [3]  Cause                          OPTIONAL
        }
-- If releaseCause is not present, the default value is the same as the ISUP cause value decimal 31.

FilteringCharacteristics ::= CHOICE {
        interval                        [0]  INTEGER,
        numberOfCalls                   [1]  INTEGER
        }
-- Indicates the severity of the filtering and the point in time when the ServiceFilteringResponse is to be sent.
-- If = interval, every interval of time the next call leads to an InitialDP and a ServiceFilteringResponse is sent to
-- the SCF. If = NumberOfCalls, every N calls the Nth call leads to an InitialDP and a ServiceFilteringResponse
-- is sent to the SCF.

FilteringCriteria ::= CHOICE {
        dialledNumber                   [0]  Digits,
        callingLineID                   [1]  Digits,
        serviceKey                      [2]  ServiceKey
        }

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

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

```
GapCriteria ::= CHOICE {
        calledPartyNumber               [0]  Digits,
        callingPartyNumber              [1]  Digits,
        gapOnService                    [2]  GapOnService
        }
-- Both calledPartyNumber and callingPartyNumber can be incomplete numbers, in the sense that a
-- limited amount of digits can be given. In that case the gapping will apply to all numbers which start
-- with the same string of digits.

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

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

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

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

InbandInfo ::= SEQUENCE {
        messageID                       [0]  MessageID,
        numberOfRepetitions             [1]  INTEGER                      DEFAULT 1,
        duration                        [2]  INTEGER                      OPTIONAL,
        interval                        [3]  INTEGER                      OPTIONAL
        }

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

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

InvokeID ::= INTEGER
-- Operation invoke identifier.

IPAvailable ::= OCTET STRING     -- defined by network operator.
-- Indicates that the resource is available.

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

IPSSPCapabilities ::= OCTET STRING        -- defined by network operator.
-- Indicates the SRF resources available at the SSP.

ISDNAccessRelatedInformation ::= OCTET STRING
-- Indicates the destination user network interface related information. Refer to the Q.763 Access
-- Transport parameter for encoding.
```

```
LegID ::= CHOICE {
        sendingSideID                   [0]  OCTET STRING,
        receivingSideID                 [1]  OCTET STRING,
        both                            [2]  SEQUENCE {
                                             sendingSideID    [0]  OCTET STRING,
                                             receivingSideID  [1]  OCTET STRING
                                             }
        }
-- Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use
-- with a choice of unilateral ID assignment or bilateral ID assignment.
-- OPTIONAL for LegID also denotes the following:
-- – when only one party exists in the call, this parameter is not needed (as no ambiguity exists);
-- – when more than one party exists in the call, one of the following alternatives applies:
--   1. LegID is present and indicates which party is concerned.
--   2. LegID is not present and a default value is assumed (e.g. calling party in the case of the
       ApplyCharging operation).
-- Choice between these two alternatives is kept a network operator option.

LocationNumber ::= OCTET STRING

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

MaximumNumberOfCounters ::= INTEGER (0..99)

MessageID ::= CHOICE {
        elementaryMessageID             [0]  INTEGER,
        text                            [1]  SEQUENCE {
                                             messageContent  [0]  IA5String,
                                             attributes      [1]  OCTET STRING OPTIONAL
                                             }
        }
-- OPTIONAL denotes network operator specific use.

MiscCallInfo ::= SEQUENCE {
        messageType                     [0]  ENUMERATED {
                                             request(0),
                                             notification(1)
                                             },
        dpAssignment                    [1]  ENUMERATED {
                                             individualLine(0),
                                             groupBased(1),
                                             officeBased(2)
                                             }  OPTIONAL
        }
-- Indicates detection point related information.

MonitorMode  ::= ENUMERATED {
        interrupted(0),
        notifyAndContinue(1),
        transparent(2)
        }
-- Indicates the event is relayed and/or processed by the SSP.

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

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

OriginalCalledPartyID ::= OCTET STRING

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

RedirectingPartyID ::= OCTET STRING

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

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

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

RequestedInformationList ::= SEQUENCE OF RequestedInformation
```

```
RequestedInformationTypeList ::= SEQUENCE OF RequestedInformationType

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

RequestedInformationType ::= ENUMERATED {
        callAttemptElapsedTime(0),
        callStopTime(1),
        callConnectedElapsedTime(2),
        calledAddress(3),
        callingAddress(4),
        bearerCapability(5)
        }

RequestedInformationValue ::= CHOICE {
        callAttemptElapsedTimeValue        [0]  INTEGER,
        callStopTimeValue                  [1]  DateAndTime,
        callConnectedElapsedTimeValue      [2]  INTEGER,
        calledAddressValue                 [3]  Digits,
        callingAddressValue                [4]  Digits,
        bearerCapabilityValue              [5]  BearerCapability
        }
-- Units for the time values are network operator specific.

ResourceID ::= CHOICE {
        lineID                             [0]  Digits,
        facilityGroupID                    [1]  FacilityGroup,
        facilityGroupMemberID              [2]  INTEGER,
        trunkGroupID                       [3]  INTEGER
        }
-- Indicates a logical identifier for the physical termination resource.

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

RouteList ::= SEQUENCE SIZE(1..3) OF OCTET STRING
-- Indicates a list of trunk groups or a route index. See Q.1214 for additional information on this item.

ScfID ::= OCTET STRING -- defined by network operator.
-- Indicates the SCF identifier.

ServiceAddressInformation ::= SEQUENCE {
        serviceKey                         [0]  ServiceKey,
        miscCallInfo                       [1]  MiscCallInfo
        }
-- Information that allows the SCF to choose the appropriate service logic with additional DP
-- information that is optional for network operator specific use.

ServiceKey ::= INTEGER
-- Information that allows the SCF to choose the appropriate service logic.

ServiceProfileIdentifier ::= OCTET STRING
-- Indicates a particular ISDN terminal. Refer to Q.932 for encoding.

TerminalType ::= ENUMERATED {
        unknown(0),
        dialPulse(1),
        dtmf(2),
        isdn(3),
        spare(16)
        }
-- Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability
-- (voice recognition, DTMF, display capability, etc.). Since present signalling systems do not convey
-- terminal type, this parameter applies only at originating or terminating local exchanges.
```

```
TimerID ::= ENUMERATED {
        tssf(0)
        -- others ffs
        }
-- Indicates the timer to be reset.

TimerValue ::= INTEGER
-- Indicates the timer value (in seconds).

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

TravellingClassMark ::= OCTET STRING (SIZE(2))
-- Indicates travelling class mark information. The value of this octet string is network operator specific.

UnavailableNetworkResource ::= ENUMERATED {
        unavailableResources(0),
        componentFailure(1),
        basicCallProcessingException(2),
        resourceStatusFailure(3),
        endUserFailure(4)
        }
-- Indicates the network resource that failed.

END
```

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

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

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

DEFINITIONS ::=

BEGIN

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

IMPORTS

-- operation types
        ActivateServiceFiltering,
        ActivityTest,
        AnalysedInformation,
        AnalyseInformation,
        ApplyCharging,
        ApplyChargingReport,
        AssistRequestInstructions,
        CallGap,
        CallInformationReport,
        CallInformationRequest,
        Cancel,
        CancelStatusReportRequest,
        CollectedInformation,
        CollectInformation,
        Connect,
        ConnectToResource,
        Continue,
        DisconnectForwardConnection,
        EstablishTemporaryConnection,
        EventNotificationCharging,
        EventReportBCSM,
        FurnishChargingInformation,
        HoldCallInNetwork,
        InitialDP,
        InitiateCallAttempt,
        OAnswer,
        OCalledPartyBusy,
        ODisconnect,
        OMidCall,
        ONoAnswer,
        OriginationAttemptAuthorized,
        PlayAnnouncement,
        PromptAndCollectUserInformation,
        Query,
        ReleaseCall,
        RequestCurrentStatusReport,
        RequestEveryStatusChangeReport,
        RequestFirstStatusMatchReport,
        RequestNotificationChargingEvent,
        RequestReportBCSMEvent,
        ResetTimer,
        RouteSelectFailure,
        SdfResponse,
        SelectFacility,
        SelectRoute,
        SendChargingInformation,
        ServiceFilteringResponse,
        SpecializedResourceReport,
        StatusReport,
        TAnswer,
        TCalledPartyBusy,
        TDisconnect,
        TermAttemptAuthorized,
        TMidCall,
        TNoAnswer,
        UpdateData
FROM IN-CS-1-Operations { ccitt recommendation q 1218 modules(0) cs-1-operations(0) version1(0) }
-- error types
        Canceled,
        CancelFailed,
        DatabaseError,
        ETCFailed,
        ImproperCallerResponse,
        InfoKeyError,
        MissingCustomerRecord,
        MissingParameter,
        ParameterOutOfRange,
        Referral,
        RequestedInfoError,
        SystemFailure,
        TaskRefused,
        UnavailableResource,
        UnexpectedComponentSequence,
        UnexpectedDataValue,
        UnexpectedParameter,
        UnknownLegID,
        UnknownResource
```

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

-- the operations are grouped by the identified ASEs.

-- SCF activation ASE

   initialDP                          InitialDP                          ::= localValue 0

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

-- Advanced BCP DP ASE
  oMidCall                            OMidCall                           ::= localValue 14
  tMidCall                            TMidCall                           ::= localValue 15

-- SCF/SRF activation of assist ASE
  assistRequestInstructions           AssistRequestInstructions          ::= localValue 16

-- Assist connection establishment ASE
  establishTemporaryConnection        EstablishTemporaryConnection       ::= localValue 17

-- Generic disconnect resource ASE
  disconnectForwardConnection         DisconnectForwardConnection        ::= localValue 18

-- Non-assisted connection establishment ASE
  connectToResource                   ConnectToResource                  ::= localValue 19

-- Connect ASE (elementary SSF function)
  connect                             Connect                            ::= localValue 20

-- Call handling ASE (elementary SSF function)
  holdCallInNetwork                   HoldCallInNetwork                  ::= localValue 21
  releaseCall                         ReleaseCall                        ::= localValue 22

-- BCSM Event handling ASE
  requestReportBCSMEvent              RequestReportBCSMEvent             ::= localValue 23
  eventReportBCSM                     EventReportBCSM                    ::= localValue 24

-- Charging Event handling ASE
  requestNotificationChargingEvent    RequestNotificationChargingEvent   ::= localValue 25
  eventNotificationCharging           EventNotificationCharging          ::= localValue 26

-- SSF call processing ASE
  collectInformation                  CollectInformation                 ::= localValue 27
  analyseInformation                  AnalyseInformation                 ::= localValue 28
  selectRoute                         SelectRoute                        ::= localValue 29
  selectFacility                      SelectFacility                     ::= localValue 30
  continueContinue                    ::= localValue 31

-- SCF call initiation ASE
  initiateCallAttempt                 InitiateCallAttempt                ::= localValue 32

-- Timer ASE
  resetTimer                          ResetTimer                         ::= localValue 33

-- Billing ASE
  furnishChargingInformation          FurnishChargingInformation         ::= localValue 34
```

```
-- Charging ASE
  applyCharging                    ApplyCharging                    ::= localValue 35
  applyChargingReport              ApplyChargingReport              ::= localValue 36

-- Status reporting ASE
  requestCurrentStatusReport       RequestCurrentStatusReport       ::= localValue 37
  requestEveryStatusChangeReport   RequestEveryStatusChangeReport   ::= localValue 38
  requestFirstStatusMatchReport    RequestFirstStatusMatchReport    ::= localValue 39
  statusReport                     StatusReport                     ::= localValue 40

-- Traffic management ASE
  callGap                          CallGap                          ::= localValue 41

-- Service management ASE
  activateServiceFiltering         ActivateServiceFiltering         ::= localValue 42
  serviceFilteringResponse         ServiceFilteringResponse         ::= localValue 43

-- Call report ASE
  callInformationReport            CallInformationReport            ::= localValue 44
  callInformationRequest           CallInformationRequest           ::= localValue 45

-- Signalling control ASE
  sendChargingInformation          SendChargingInformation          ::= localValue 46

-- Specialized resource control ASE
  playAnnouncement                 PlayAnnouncement                 ::= localValue 47
  promptAndCollectUserInformation  PromptAndCollectUserInformation  ::= localValue 48
  specializedResourceReport        SpecializedResourceReport        ::= localValue 49

-- User data manipulation ASE
  query                            Query                            ::= localValue 50
  sdfResponse                      SdfResponse                      ::= localValue 51
  updateData                       UpdateData                       ::= localValue 52

-- Cancel ASE
  cancel                           Cancel                           ::= localValue 53
  cancelStatusReportRequest        CancelStatusReportRequest        ::= localValue 54

-- Activity Test ASE
  activityTest                     ActivityTest                     ::= localValue 55

-- ERROR codes
  canceled                         Canceled                         ::= localValue 0
  cancelFailed                     CancelFailed                     ::= localValue 1
  databaseError                    DatabaseError                    ::= localValue 2
  eTCFailed                        ETCFailed                        ::= localValue 3
  improperCallerResponse           ImproperCallerResponse           ::= localValue 4
  infoKeyError                     InfoKeyError                     ::= localValue 5
  missingCustomerRecord            MissingCustomerRecord            ::= localValue 6
  missingParameter                 MissingParameter                 ::= localValue 7
  parameterOutOfRange              ParameterOutOfRange              ::= localValue 8
  referral                         Referral                         ::= localValue 9
  requestedInfoError               RequestedInfoError               ::= localValue 10
  systemFailure                    SystemFailure                    ::= localValue 11
  taskRefused                      TaskRefused                      ::= localValue 12
  unavailableResource              UnavailableResource              ::= localValue 13
  unexpectedComponentSequence      UnexpectedComponentSequence      ::= localValue 14
  unexpectedDataValue              UnexpectedDataValue              ::= localValue 15
  unexpectedParameter              UnexpectedParameter              ::= localValue 16
  unknownLegID                     UnknownLegID                     ::= localValue 17
  unknownResource                  UnknownResource                  ::= localValue 18

END
```

3 Procedures

3.1 Definition of procedures and entities

3.1.1 SSF application entity procedures

3.1.1.1 General

This subclause provides the definition of the SSF application entity (AE) procedures related to the SSP – SCP interface. The procedures are based on the use of Common Channel Signalling System No. 7 (CCSS No. 7); other signalling systems can be used (e.g. DSS 1-layer 3).

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

The AE, following the architecture defined in the Recommendations Q.700, Q.771 and Q.1400, includes 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, 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.

3.1.1.2 Model and interfaces

The functional model of the AE-SSF is shown in Figure 9; the ASEs interface to TCAP to communicate with the SCF, and interface to the call control function (CCF) and the maintenance functions already defined for switching systems. The scope of this Recommendation is limited to the shaded area in Figure 9.

The interfaces shown in Figure 9 use the TC-user ASE primitives specified in Recommendation Q.771 [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 2.

3.1.1.3 Relations between SSF FSM and the CCF and maintenance functions

The primitive interface between the SSF FSM and the CCF/maintenance functions is an internal interface and is not subject to standardization in CS-1. Nevertheless this interface should be in line with the BCSM defined in 4.2.1.2/Q.1214.

The relationship between the BCSM and the SSF FSM may be described as follows for the case of a call/attempt initiated by an end user, and the case of a call/attempt initiated by IN service logic:

- When a call/attempt is initiated by an end user and processed at an exchange, an instance of a BCSM is created. As the BCSM proceeds, it encounters detection points (DPs, see 4.2/Q.1214). If a DP is armed as a Trigger DP (TDP) an instance of an SSF FSM is created.

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

The management functions related to the execution of operations received from the SCF are executed by the SSF management entity (SSME). The SSME interfaces the different SSF FSMs and the functional entity access manager (FEAM). Figure 10 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 SSF management entity (SSME);

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

The SSME maintains the dialogues with the SCF, and SRF on behalf of all instances of the SSF finite state machine (FSM). These instances of the SSF FSM occur concurrently and asynchronously as calls occur, which explains the need for a single entity that performs the task of creation, invocation, and maintenance of the SSF FSMs. In particular, the SSME 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. For example, the SSME provides non-call associated treatment due to changes in service filtering, Call Gapping, or resource monitoring status, and also provides resource status messages to the SCF. Therefore, the SSME separates the SSF FSM from the Call Gapping, service filtering and resource monitoring functions.

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

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

3.1.1.4 SSF management entity (SSME) state diagram

The SSF management entity (SSME) state diagram is described in Figure 11.

The SSME is independent of the individual SSF FSMs.

The non-call associated treatment state is entered from the IdleManagement state when one of the following non-call associated operations is received (transition em1):

RequestCurrentStatusReport
RequestEveryStatusChangeReport
RequestFirstStatusMatchReport,
ActivateServiceFiltering
CallGap
Activity Test

The above operations may be received inside as well as outside a call context transaction.

During this state the following events can occur:

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

– given that service filtering is active and the service filtering duration expires: the SSME should move to the IdleManagement state(transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3)

- given that status report is active, the SSF needs to send a StatusReport operation for the status monitoring type "continuous monitor:" the SSF remains in this state (transition em3);

- given that status report is active, the SSF needs to send a StatusReport operation for the status monitoring type "monitor for the first match:" the SSF transits to IdleManagement state (transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3);

- given that status report is active, the cancel (StatusReport) operation is received by the SSF or the status report duration expires, for the status monitoring types "Monitor for the first match" or "continuous monitor:" the SSME should move to the IdleManagement state (transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3);

- given that status report is active, the SSF sends the StatusReport operation, for status monitoring type "poll resource status:" the SSME should move to the IdleManagement state (transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3);

- if call gap related duration timer expires, the SSME should move to the IdleManagement state (transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3);

- given that call gap/service filtering is active, another call gap/Activate Service Filtering operation could be received by the SSF, which has the same gapping/filtering criteria: the second "filter" or "gap" replace the first one (transition em3) unless the duration timer value is equal to zero, in which case the SSF should move to the IdleManagement state (transition em2) unless there is another management activity in progress, in which case the SSF should remain in the same state (transition em3).

All other operations have no effect on the state (transition em3); the operations are passed to the relevant SSF FSM.

3.1.1.5 SSF state transition diagram

Figure 12 shows the state diagram of the SSF part of the SSP during the processing of an IN call/attempt.

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

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

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

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

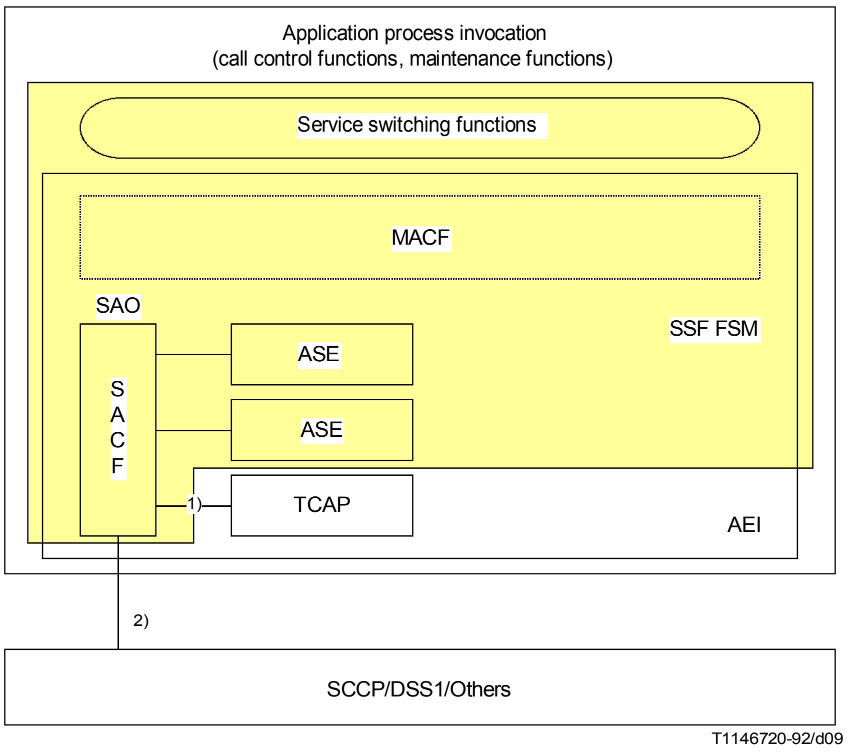
- The SSF examines subsequent operations in the sequence. As long as sequential execution of these operations would leave the FSM in the same state, it will execute them (e.g. RequestReportBCSMEvent). If a subsequent operation causes a transition out of the state then the following operations should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

  The SSF receives the operations FurnishChargingInformation, ConnectToResource, and PlayAnnouncement in a component sequence inside a single TCAP message. Upon receipt of this message, these operations are executed up to and including ConnectToResource while the SSF is in the waiting for instruction state. As the ConnectToResource operation is executed (and when, or after the FurnishChargingInformation operation has been completed), the SSF FSM will transition to the waiting for end of user interaction state. The PlayAnnouncement operation is relayed to the SRF while the SSF is in waiting for end of user interaction state.

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

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

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


Application process invocation
(call control functions, maintenance functions)
Service switching functions
MACF
SAO
SSF FSM
ASE
ASE
SACF
1)
TCAP
AEI
2)
SCCP/DSS1/Others
T1146720-92/d09

1) TC-Primitives.
2) N-Primitives.

AEI	Application entity invocation
SSF	Service switching functions
FSM	Finite state machine
MACF	Multiple association control function
SACF	Single association control function
SAO	Single association object

NOTE – SSF FSM includes several finite state machines.

FIGURE 9/Q.1218

Functional model of SSF AE

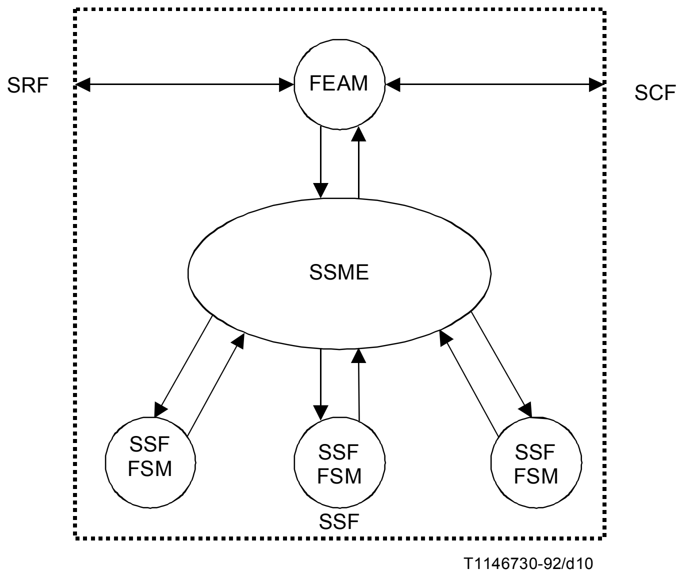


FIGURE 10/Q.1218

SSF Interfaces

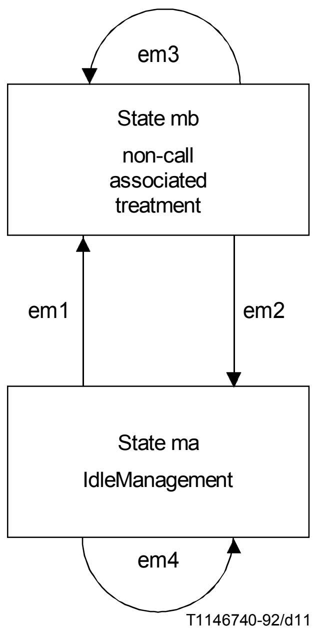
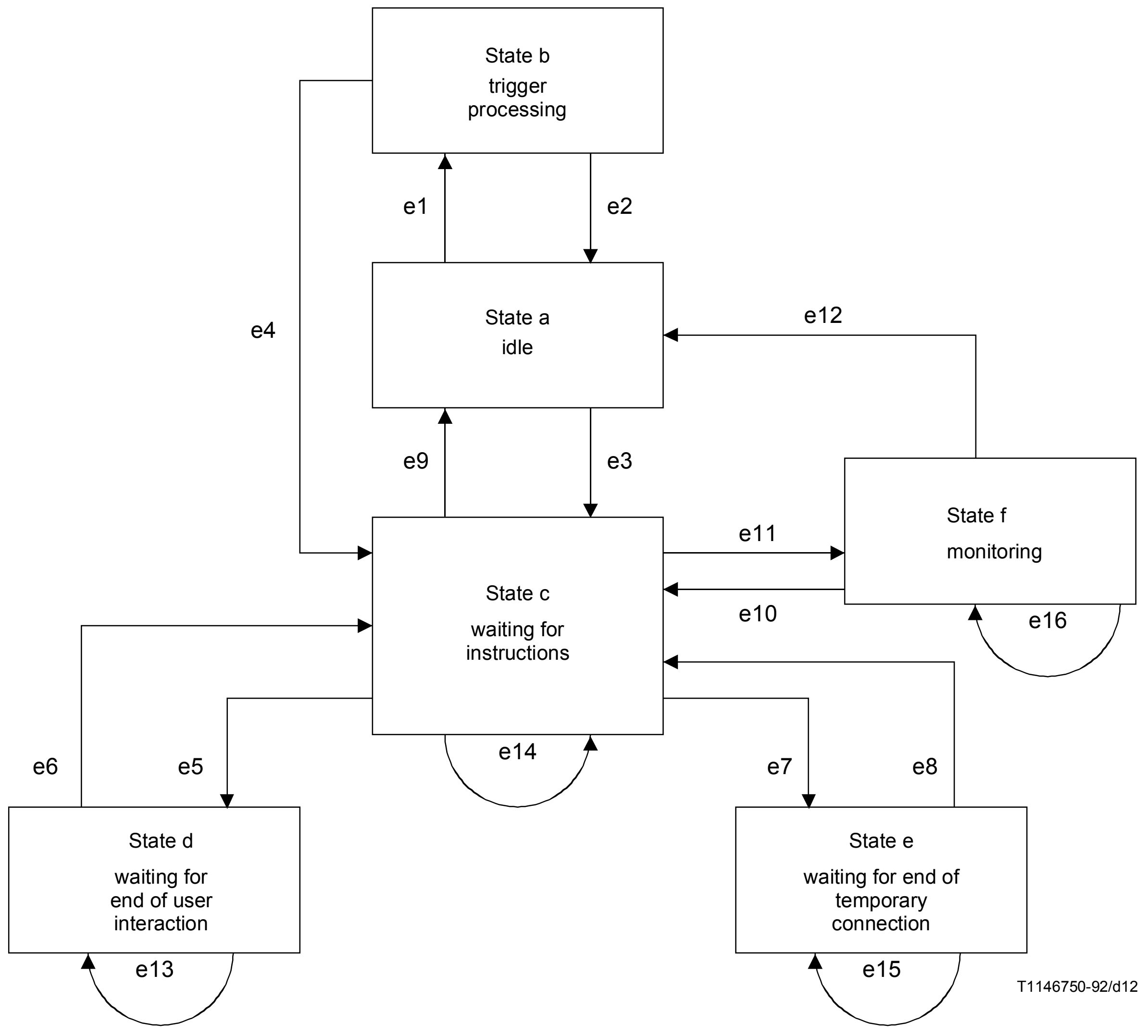


FIGURE 11/Q.1218

SSF management entity (SSME) state diagran



NOTE – The abandon and disconnect transition is not shown.

FIGURE 12/Q.1218

SSF finite state machine

In any state (except Idle), if the calling party abandons the call before it is answered (i.e. before the Active PIC in the BCSM), then the SSF FSM should instruct the CCF to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows[1]:

– If the Abandon DP is not armed and there is no call information request pending, then transition to the idle state.

– If the Abandon DP is not armed and there is a call information request pending, send a CallInformationReport, then transition to the idle state.

[1] Other pending requests that should be treated in the same way as the CallInformationRequest are the RequestNotificationChargingEvent and/or the Apply Charging when the "SendCalculationToSCPIndication" parameter is set to True.

- If the Abandon DP is armed as an EDP-R, send an EventReportBCSM or a DP specific operation, then transition to the waiting for instructions state (whether or not there is a pending call information request).
- If the Abandon DP is armed as an EDP-N and there is no call information request pending, send an EventReportBCSM or a DP specific operation, then transition to the idle state.
- If the Abandon DP is armed as an EDP-N and there is a call information request pending, send an EventReportBCSM or a DP specific operation and a CallInformationReport, then transition to the idle state.

In any state (except idle), if a call party disconnects from a stable call (i.e. from the Active PIC in the BCSM), then the SSF FSM should process this event as follows[2]:

- If the disconnect DP is not armed and there is no call information request pending, transition to the Idle state if there are no remaining call parties.
- If the disconnect DP is not armed and there is a call information request pending, send a CallInformationReport and transition to the Idle state.
- If the disconnect DP is armed as an EDP-R, send an EventReportBCSM or a DP specific operation, then transition to the waiting for instructions state (whether or not there is a pending call information request). Then if there are no remaining parties, continue clearing the call.
- If the disconnect DP is armed as an EDP-N and there is no call information request pending, send an EventReportBCSM or a DP specific operation, then transition to the Idle state.
- If the disconnect DP is armed as an EDP-N and there is a call information request pending, send an EventReportBCSM or a DP specific operation and a CallInformationReport, then transition to the idle state.

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

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

- when the SSF sends an InitialDP, DP specific or AssistRequestInstructions operation (Ref. 3.1.1.5.3 State c: "waiting for instructions"). While waiting for the first response from the SCF, the timer T_{SSF} can be reset only once by a reset timer operation. Subsequent to the first response, the timer can be reset any number of times.
- when the SSF receives a HoldCallInNetwork operation (Ref. 3.1.1.5.3 State c: "waiting for instructions"). In this case the SCF may reset T_{SSF} using the reset timer 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 (Ref. 3.1.1.5.4 and 3.1.1.5.5). In these cases the SCF may reset T_{SSF} using the reset timer operation any number of times.

In the three above cases T_{SSF} may respectively have three different values as defined by the application.

When receiving or sending any operation which is different from the above, the SSF should reset T_{SSF}. In the "monitoring" state (Ref. 3.1.1.5.6) T_{SSF} is not used.

On expiration of T_{SSF} the SSF FSM transitions to the idle state, aborts the interaction with the SCF and the CCF progresses the BCSM if possible.

[2] Other pending requests that should be treated in the same way as the CallInformationRequest are the RequestNotificationChargingEvent and/or the Apply Charging when the "SendCalculationToSCPIndication" parameter is set to True.

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

e1	TDP encountered
e2	Trigger fail
e3	Initiate call received
e4	Trigger detected
e5	User interaction requested
e6	User interaction ended
e7	Temporary connection created
e8	Temporary connection ended
e9	Idle return from wait for instruction
e10	EDP_R encountered
e11	Routing instruction received
e12	EDP_N last encountered
e13	Waiting for end of user interaction state no change
e14	Waiting for instruction state no change
e15	Waiting for end of temporary connection state no change
e16	Monitoring state no change
e17	Abandon (from any state) (not shown in the SSF state diagram)
e18	Disconnect (from any state) (not shown in the SSF state diagram)
e19	Non-call associated treatment from any state (not shown in the SSF state diagram)

The SSF state diagram contains the following states:

State a	Idle
State b	Trigger processing
State c	Waiting for instructions
State d	Waiting for end of user interaction
State e	Waiting for end of temporary connection
State f	Monitoring

3.1.1.5.1 State a: "Idle"

The SSF FSM enters the idle state under a variety of conditions, as described below.

The SSF FSM enters the idle state when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state.

The SSF FSM enters the idle state when DP processing fails in the trigger processing state (transition e2).

The SSF FSM enters the idle state when one of the following occurs:

- when the call is abandoned or all call parties disconnect in any other state under the conditions identified in 3.1.1.5;
- when an initiate call, connect, or proceed call processing operation is processed in the waiting for instructions state, and no EDPs are armed and monitoring is not active for charging events (transition e9);
- when the application timer T_{SSF} expires in the waiting for instructions state (transition e9);

- when a ReleaseCall operation is processed in waiting for instructions (transition e9) or monitoring (transition e12);
- when the last EDP-N is encountered in the monitoring state, and there are no EDP-Rs armed and no monitoring is active for charging events (transition e12);
- when the last charging event is encountered in the monitoring state, and there are no EDPs armed (transition e12).

When transitioning to the idle state, if there is a call information request pending (see 3.1.1.5), the SSF sends a CallInformationReport operation to the SCF before returning to idle. Once in the idle state, if status reporting is still active the SSF deactivates it, any outstanding responses to send to the SCF are discarded.

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

- indication from the CCF that an armed TDP is encountered related to a possible IN call/service attempt: in this case the SSF FSM moves to the state trigger processing (transition e1).
- a message related to a new transaction containing an Initiate CallAttempt operation is received from the SCF: in this case the SSF FSM moves to the state waiting for instructions (transition e3).

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

3.1.1.5.2 State b: "trigger processing"

Following a trigger detection related to an armed TDP in the BCSM, the SSF FSM is activated and moves from the idle state to the trigger processing state (transition e1).

In this state, the SSF/CCF should:

- perform the DP processing actions specified in 4.2.2.5/Q.1214
  a) check if call gapping or service filtering mechanisms are active;
  b) check for SCF accessibility;
  c) determine if DP criteria are met;
  d) handle service feature interactions;
- collect and verify necessary parameters for sending an InitialDetectionPoint or a DP-specific[3)] operation to the SCF:
  1) if successful and the DP is a TDP-N, send a generic InitialDetectionPoint or DP-specific operation[3)] to the SCF, as determined from DP processing, and transition back to the idle state (transition e2);
  2) if successful and the DP is a TDP-R, send a generic InitialDetectionPoint or DP-specific operation[3)] to the SCF, as determined from DP processing, and transition to the waiting for instructions state (transition e4);
  3) if DP processing fails, return to the idle state (transition e2). DP processing fails in the following cases:
     - if CallGapping is in effect: the SSF FSM will instruct the CCF to terminate the call with the appropriate treatment;
     - if service filtering is in effect: the call is counted (if required) and the SSF FSM instructs the CCF to terminate the call with the appropriate treatment;

[3)] *DP specific* operations are the following (ref. 2): TAnswer, TDisconnect, TermAttemptAuthorized, TMidCall, TNoAnswer, AnalysedInformation, TCalledPartyBusy, CollectedInformation, OAnswer, OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer, OriginationAttemptAuthorized, RouteSelectFailure.

- if a trigger criteria match is not found: the SSF FSM returns call control to the CCF;
- if the call is abandoned: the SSF returns call control to the CCF and continues processing as described in 3.1.1.5;
- if there is insufficient information to proceed (e.g. applies to open numbering plans): the SSF FSM instructs the CCF to terminate the call with a standard terminating treatment;
- if the destination SCF is not accessible: the SSF FSM will instruct the CCF to route the call if possible (e.g. default routing to a terminating announcement);
- if there is an existing control relationship for the call: the SSF returns call control to the CCF.

3.1.1.5.3 State c: "waiting for instructions"

This state is entered from either the trigger processing state, as indicated above (transition e4), or directly from the idle state on receipt at the SSF of an OPEN primitive containing an "initiate call" operation from the SCF (transition e3). In the latter case, the SSF moves to the waiting for instructions state immediately and then proceeds according to the contents of the "initiate call" operation and any other received operations.

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

During this state the following events can occur:

- The user dials additional digits (applies for open-ended numbering plans): the SSF should store the additional digits dialled by the user.
- The user abandons or disconnects (see 3.1.1.5).
- The application Timer T_{SSF} expires: the SSF FSM moves to the idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: The SSF FSM acts according to the operation received as described below.

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

HoldCallInNetwork
RequestReportBCSMEvent,
RequestNotificationChargingEvent
ResetTimer
FurnishChargingInformation
ApplyCharging
CallInformationRequest
SendChargingInformation

The following operations may be received from the SCF and processed by the SSF, causing a state transition to waiting for end of user interaction state (transition e5):

ConnectToResource

The following operations may be received from the SCF and processed by the SSF, causing a state transition to waiting for end of temporary connection state (transition e7):

EstablishTemporaryConnection

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

Connect
CollectInformation
AnalyseInformation
SelectRoute
SelectFacility
Continue
ReleaseCall

Initiate CallAttempt operation, if received in idle state, should be processed in waiting for instructions state (transition e3).

When processing the above operations, any necessary call handling information is provided to the call control function (CCF).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5 .

Call Abandon/Disconnect should be processed in accordance with the general rules in 3.1.1.5 .

3.1.1.5.4 State d: "waiting for end of user interaction"

The SSF enters this state from the waiting for instructions state (transition e5) on the reception of one of the following operations:

ConnectToResource

During this state the following events can occur:

- A valid SCF-SRF operation [i.e. play announcement, prompt & collect user information, and cancel (announcement)] for relaying is received and is correct, the operation is transferred to the SRF for execution. The SSF FSM remains in the waiting for end of user interaction state (transition e13).
- A valid SRF-SCF operation (i.e. SpecializedResourceReport and ReturnResult from prompt and collect user information) for relaying is received and is correct, the operation is transferred to the SCF. The SSF FSM remains in the waiting for end of user interaction state (transition e13).
- The application timer T_{SSF} expires: the SSF FSM moves to the idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: The SSF FSM acts according to the operation received as described below.

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

RequestReportBCSMEvent,
RequestNotificationChargingEvent
ResetTimer
FurnishChargingInformation
ApplyCharging
CallInformationRequest
Send Charging Information

The DisconnectForwardConnection operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the waiting for instructions state. The disconnection is not transferred to the other party (transition e6).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

Call abandon should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.5.5 State e: "waiting for end of temporary connection"

The SSF enters this state from the waiting for instructions state (transition e7) upon receiving an EstablishTemporaryConnection operation.

The call is routed to the assisting SSF/SRF and call handling is suspended while waiting for the end of the assisting procedure. The timer T_{SSF} is active in this state.

During this state the following events can occur:

– The reception of a DisconnectForwardConnection. In this case, the SSF moves to the waiting for instructions state (transition e8).

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

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

RequestReportBCSMEvent
RequestNotificationChargingEvent
ResetTimer
FurnishChargingInformation
ApplyCharging
CallInformationRequest
SendChargingInformation

The DisconnectForwardConnection operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the waiting for instructions state. The disconnection is not transferred to the other party (transition e8).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

Call abandon should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.5.6 State f: "Monitoring"

The SSF enters this state from the Wait for instructions state (transition e11) upon receiving a Connect, CollectInformation, AnalyseInformation, SelectRoute, SelectFacility, Continue, ReleaseCall operation or processing an Initiate CallAttempt operation when one or more EDPs are armed or/and there is call information request pending (see 3.1.1.5).

In this state the timer T_{SSF} is not used; i.e., the expiration of T_{SSF} does not have any impact on the SSF FSM.

During this state the following events can occur:

– An EDP-N should be reported to the SCF by sending an Event ReportBCSM operation or a DP specific operation; the SSF FSM should remain in the monitoring state (transition e16) if one or more EDPs are armed or there is a call information request pending. The SSF FSM should move to the idle state (transition e12) if there are no remaining EDPs armed or there is no call information request pending.

– An EDP-R should be reported to the SCF by sending an Event ReportBCSM operation or a DP specific operation; the SSF FSM should move to the wait for instructions state (transition e10).

- The receipt of an END or ABORT primitive from TCAP has no effect on the call; the call may continue or be completed with the information available. In this case, the SSF FSM transitions to the idle state (transition e12), disassociating the SSF FSM from the call.
- An operation is received from the SCF: The SSF FSM acts according to the operation received as described below.

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

RequestReportBCSMEvent
RequestNotificationChargingEvent
CallInformationRequest

The ReleaseCall operation may be received from the SCF and processed by the SSF, causing a state transition to the idle state (transition e12). If there is a call information request pending, the SSF sends a CallInformationReport to the SCF.

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

Call Abandon/Disconnect should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.6 Assisting/Handoff SSF FSM

This subclause describes the SSF FSM related to the Assisting/Handoff SSF. The assisting SSF is structured as defined in 3.1.1.1 through 3.1.1.5. The Handoff FSM for CS-1 applies only to the case where final treatment is to be applied.

The Assisting/Handoff SSF state diagram contains the following transitions (events) (see Figure 13):

ea1	Assist/Handoff detected
ea2	Assist/Handoff fail
ea3	User interaction requested
ea4	User interaction ended
ea5	Waiting for instruction state no change
ea6	Waiting for end of user interaction state no change

The Assisting/Handoff SSF state diagram contains the following states:

State a	Idle
State b	Waiting for instructions
State c	Waiting for end of user interaction

3.1.1.6.1 State a:"Idle"

The SSF FSM enters the idle state when one of the following occurs:

- when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state
- given a temporary connection between an upstream SSF and the Assisting SSF, when a DisconnectForwardConnection indication is received from the upstream SSF; (transition ea2).

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

The Assisting SSF FSM transitions from the idle state to the waiting for instructions state on receipt of an assist indication at the assisting SSF from another SSF (transition ea1).

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

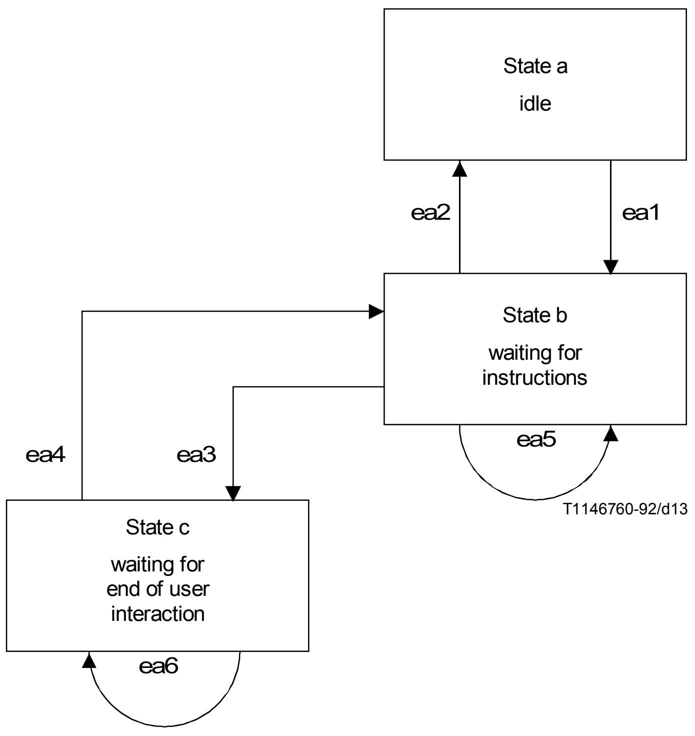


FIGURE 13/Q.1218

Assisting/Handoff SSF finite state machine

3.1.1.6.2 State b: "waiting for instructions"

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

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

During this state the following events can occur:

- The application timer T_{SSF} expires: the Assisting SSF FSM moves to the idle state (transition ea2) and the expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: The SSF FSM acts according to the operation received as described below.
- A bearer channel disconnect is received and the FSM moves to the idle state (transition ea2).

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

ResetTimer
FurnishChargingInformation
ApplyCharging
SendChargingInformation

The following operations can be received from the SCF and processed by the Assisting SSF, causing a state transition to waiting for end of user interaction state (transition ea3):

ConnectToResource

The following operations can be received from the SCF and processed by the Assisting SSF, causing a state transition to idle state (transition ea2):

ReleaseCall

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

Note that multiple Handoff procedures are not covered by this Recommendation.

3.1.1.6.3 State c: "waiting for end of user interaction"

The Assisting SSF enters this state from the waiting for instructions state (transition ea3) on the reception of one of the following operations:

ConnectToResource

During this state the following events can occur:

- A valid SCF-SRF operation [i.e. Play announcement, prompt & collect user information, and cancel (announcement)] for relaying is received and is correct, the operation is transferred to the SRF for execution. The SSF FSM remains in the waiting for end of user interaction state (transition ea6).
- When the SRF indicates to the SSF the end of user interaction by initiating disconnection the SSF FSM returns to the waiting for instructions state (transition ea4).
- The application timer T_{SSF} expires: the SSF FSM moves to the idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: The SSF FSM acts according to the operation received as described below.

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 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 3.1.1.5.

3.1.2 SCF application entity procedures

3.1.2.1 General

This subclause provides the definition of the SCF application entity (AE) procedures related to the SCF-SSF/SRF/SDF interface. 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 TCAP (Transaction capabilities application part) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF and MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

The procedure may equally be used with other message-based signalling systems supporting the application layer structures defined. By no means is this text intended to dictate any limitations to service logic programs.

3.1.2.2 Model and interfaces

The functional model of the AE-SCF is shown in Figure 14; the ASEs interface with supporting protocol layers to communicate with the SSF, SRF and SDF, and interface to the service logic programs and maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 14.

The interfaces shown in Figure 14 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and in N-primitives specified Recommendation Q.711 [interface (2)]. The operations and parameters of intelligent network application protocol (INAP) are defined in 2.

3.1.2.3 Relationship between the SCF FSM and SLPs/Maintenance Functions

The primitive interface between the SCF FSM and the service logic programs/maintenance functions is an internal interface and is not a subject for standardization in CS-1.

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

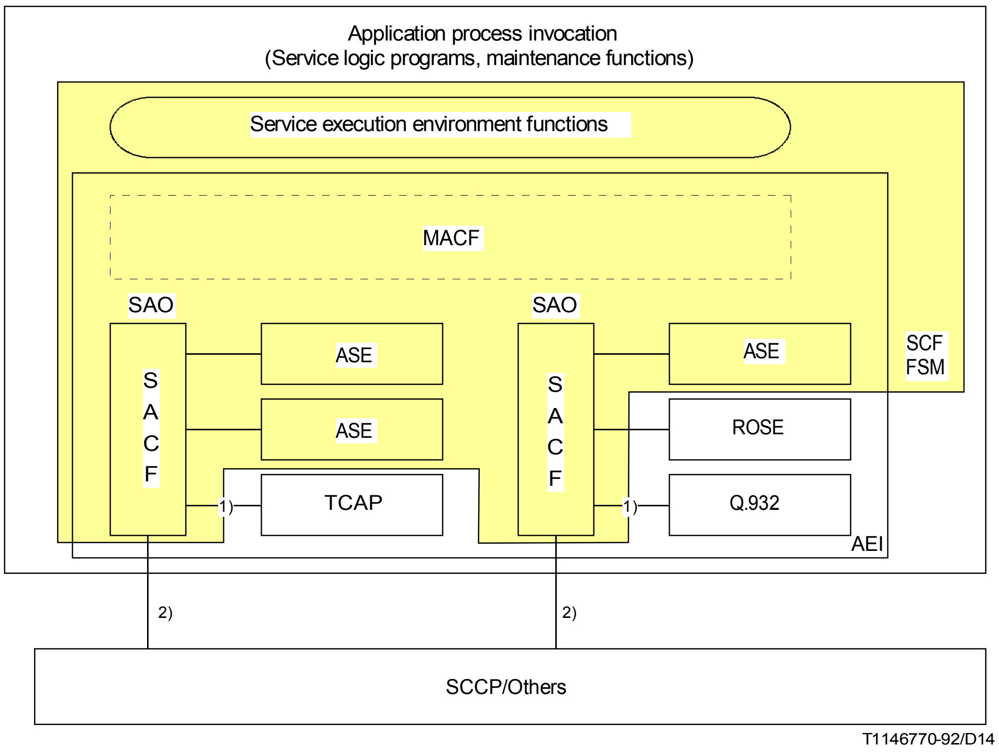
- If a request for IN call processing is received from the SSF, an instance of an SCF state model (SCSM) is created, and the relevant service logic program is invoked.
- When initiation of a call is requested from service logic, an instance of the SCSM is created.

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 service logic program of events as needed.

The management functions related to the execution of operations received from the SCF are executed by the SCF management entity (SCME). The SCME interfaces different SCF call state models (SCSMs) and the functional entity access manager (FEAM). Figure 15 shows the SCF FSM structure.

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

The procedural model associates an SCSM with each query from the SSF. The SCSM maintains dialogues with the SSF, SRF, and SDF on behalf of service logic.


Application process invocation
(Service logic programs, maintenance functions)
Service execution environment functions
MACF
SAO
SACF
ASE
ASE
TCAP
SAO
SACF
ASE
ROSE
Q.932
SCF FSM
AEI
1)
1)
2)
2)
SCCP/Others
T1146770-92/D14

AEI	Application entity invocation
SCF	Service control functions
FSM	Finite state machine
MACF	Multiple association control function
SACF	Single association control function
SAO	Single association object

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

NOTE – The SCF FSM includes several finite state machines.

FIGURE 14/Q.1218

Functional model of SCF AE

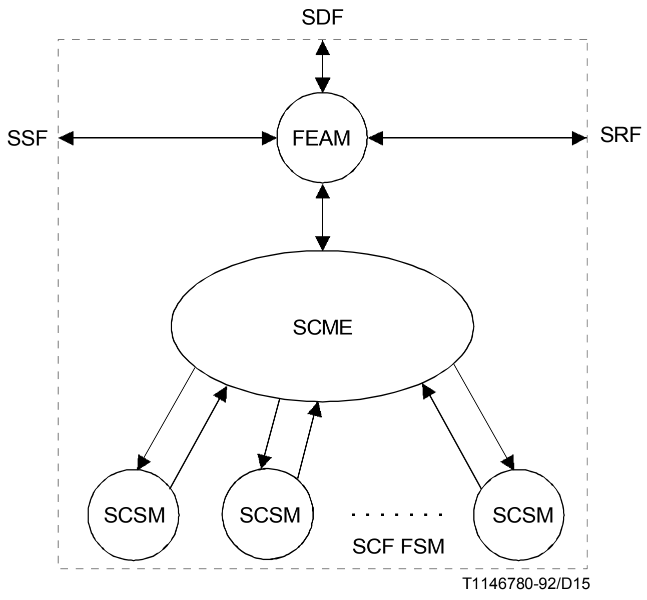
Multiple requests may be executed concurrently and asynchronously by the SCF, which explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SCF FSM objects. This entity is called the SCF management entity (SCME). In addition to the above tasks, the SCME maintains the dialogues with the SSF, SDF, and SRF on behalf of all instances of the SCSM. In particular, the SCME:

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 all asynchronous (with call processing) activities (one such activity is flow control). It is the SCME's responsibility 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; and

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



FEAM	Functional entity access manager
SCME	SCF management entity
SCSM	SCF call state model

FIGURE 15/Q.1218

SCF FSM structure

Finally, the functional entity access manager (FEAM) relieves the SCME of low-level interface functions. The FEAM's functions include:

1) Establishing and maintaining the interfaces to the SSF, SRF and SDF;

2) Passing (and queueing when necessary) the messages received from the SSF, SRF, and SDF to the SCME; and

3) Formatting, queueing (when necessary), and sending messages received from the SCME to the SSF, SRF, and SDF.

Note that although the SCSM includes a state and procedures concerning queue management, this type of resource management only represents one way of managing network call queues. Another alternative is to let the SSF/CCF manage call queues; however, the technical details of how the SSF/CCF performs queue management is beyond the scope of IN. As such, the resource control object (see 3.1.2.4.5) and the SCSM's queueing state (State 2.2), along with its relevant sub-states, events and procedures, are only required and applicable in the case where queue management is performed in the SCF.

3.1.2.4 Partial SCF management entity (SCME) state transition diagram

The two key parts of SCF management entity (SCME) state diagram are described in Figures 16 and 17.

The SCME handles the following operations:

– Request current status report;

– Request every status change report;

– Request first status match report;

- Cancel status report (including the Invoke ID previously used for request first status match report or request every state change report operation);
- Status report;
- Activate service filtering;
- Service filtering report;
- Call gap; and
- Activity test.

The issuing of the call gap and activity test operations does not cause state transitions in the SCME. The procedures for the rest of the above operations are described below.

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

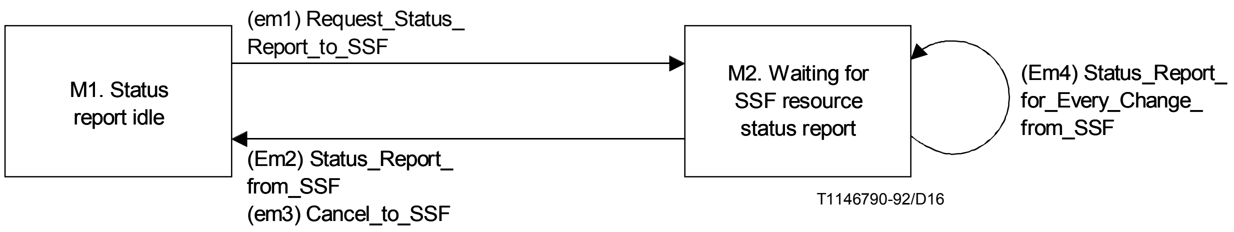

M1. Status report idle
(em1) Request_Status_ Report_to_SSF
M2. Waiting for SSF resource status report
(Em4) Status_Report_ for_Every_Change_ from_SSF
(Em2) Status_Report_ from_SSF
(em3) Cancel_to_SSF
T1146790-92/D16

FIGURE 16/Q.1218

The status report FSM in the SCME

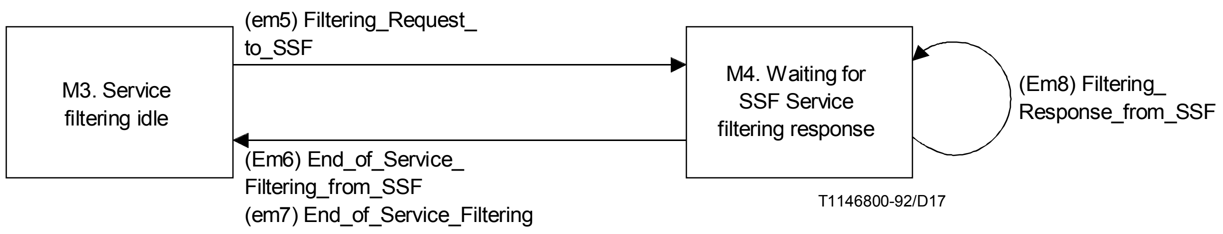

M3. Service filtering idle
(em5) Filtering_Request_ to_SSF
M4. Waiting for SSF Service filtering response
(Em8) Filtering_ Response_from_SSF
(Em6) End_of_Service_ Filtering_from_SSF
(em7) End_of_Service_Filtering
T1146800-92/D17

FIGURE 17/Q.1218

The service filtering FSM in the SCME

3.1.2.4.1 State M1: “Status report idle”

The following event[4)] is considered in this state:

- (em1) Request_Status_Report_to_SSF: This is an internal event, caused by a decision to transmit one of the following operations:
  - Request current status report;
  - Request first status match report;
  - Request every status change report.

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

3.1.2.4.2 State M2: “Waiting for SSF response status report”

The following events are considered in this state:

- (Em2) Status_Report_from_SSF: This is an external event, caused by the reception of the response to the request current status report or request first status match report operation previously issued to the SSF. This event causes a transition out of this state to state M1 status report idle;
- (em3) Cancel_to_SSF: This is an internal event, caused by the service logic’s need to end status monitoring of the resources in the SSF, and by transmission of cancel status report operation to the SSF. This event takes place only for previously issued request first status match report or request every status change report operations. This event causes a transition to state M1 status report idle; and
- (Em4) Status_Report_for_Every_Change_from_SSF: This is an external event, caused by reception of the response to the request every status change report operation previously issued to the SSF. This event does not cause a transition out of this state, so the SCSM still remains in state M2, waiting for SSF resource status report.

3.1.2.4.3 State M3: “Service filtering idle”

The following event is considered in this state:

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

3.1.2.4.4 State M4: “Waiting for SSF service filtering response”

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

- (Em6) End_of_Service_Filtering_Response_from_SSF: This is an external event, caused by reception of the response to the request service filtering previously issued to the SSF at the end of the service filtering duration. This event causes a transition out of this state to state M3, service filtering idle;
- (em7) End_of_Service_Filtering: This is an internal event, caused by the expiration of service filtering duration timer in the SCF. This event causes a transition to state M3, service filtering idle.
- (Em8) Filtering_Response_from_SSF: This is an external event, caused by reception of the response to the request service filtering operation previously issued to the SSF. This event does not cause a transition out of this state, and the SCSM remains in state M4, waiting for SSF service filtering response.

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

3.1.2.4.5 The Resource Control Object

The resource control object (RCO) is part of the SCF management entity that controls data relevant to resource information.

---

4) All events are enumerated, and the number of an event is prefixed with either the letter “E” (for external events) or “e” (for internal ones) and included in parentheses in the beginning of the event name.

The 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.

3.1.2.5 The SCSM

Figure 18 shows the general state diagram of the SCSM as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses. Each state, except idle, SDF request idle and waiting for SDF response, has internal sub-FSMs composed of the sub-states.

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

In every state, if there is an error in a received operation, the service logic program and the maintenance functions are informed, and the SCSM remains in the same state. Depending on the class of the operation, the error can be reported to the SSF, SRF, or SDF (see Recommendation Q.774).

It also holds that, in every state, if the SCSM is informed that the dialogue with the SSF is terminated, then it informs the service logic program and returns to the idle state. In this case, all resources allocated to that call, including those required for relevant dialogues with the other functions, must be de-allocated. To simplify the diagram, such transitions are not demonstrated in the figures.

When the service logic program requests call information, the SCSM transmits the call information request operation to the SSF, and then call information report is outstanding.

The cancel operation (for call information request) can only be sent to the SSF when call information report is outstanding.

In any state (except idle), the SCSM may receive the call information report[5)] operation from the SSF, when the call information report is outstanding.

From any state (except idle), if call information report is outstanding and the service logic program indicates that the processing has been completed, the SCSM remains in the same state until it receives the call information report operation.

The general rules for one or a sequence of components sent in one or more TCAP messages, which may include a single operation or multiple operations, are specified in 3.1.1.5 SSF state transition diagram (they are not described here).

[5)] Other pending requests that should be treated in the same way as the call information request are the request notification charging event and/or the apply charging when the "Send Calculation to SCF Indication" parameter is set to True.

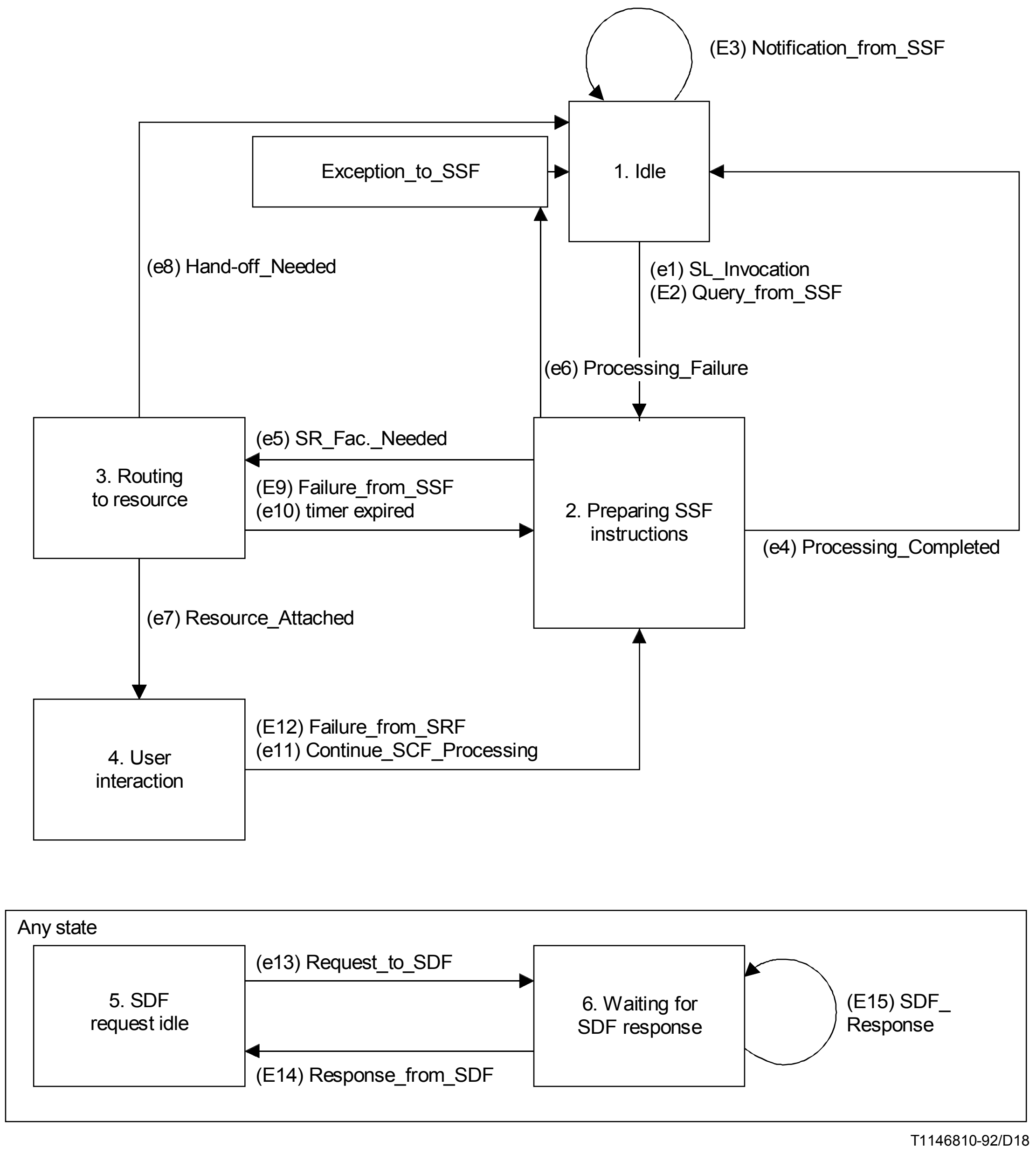


FIGURE 18/Q.1218

The SCSM finite state machine

The SCSM has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to reset the timer T_{SSF}, which is used to prevent excessive call suspension time and to guard the association between the SSF and the SCF.

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

– when the SCF receives an initial DP, DP specific request instructions or assist request instructions operation. (See 3.1.2.5.2.1 state 2.1: "Preparing SSF instructions", and 3.1.2.5.2.2.1 state 2.2.1: "Preparing SSF instructions".) In this case, this timer is reset when the first request, other than reset timer operation, is sent to the SSF. On the expiration of $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} once, using the reset timer operation, and also reset $T_{SCF\text{-}SSF}$. On second expiration of $T_{SCF\text{-}SSF}$, the SCSM informs service logic program and the maintenance functions, and the SCSM transits to the idle state;

– when the SCF sends a hold call in network operation. (See 3.1.2.5.2.2.2 state 2.2.2: "queueing".) In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} using the reset timer operation any number of timers; and

- when the SCF enters the “Waiting For Assist Request Instructions” state or the “user interaction” state (See 3.1.2.5.3.2 and 3.1.2.5.4.). In these cases, on the expiration of $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} using the reset timer operation any number of times.

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

When receiving or sending any other operation, the SCF should reset $T_{SCF\text{-}SSF}$. In the “Waiting For Notification Or Request” state (see 3.1.2.5.2.3), $T_{SCF\text{-}SSF}$ is not used.

The SCSM also has an application timer, $T_{ASSIST/HAND\text{-}OFF}$, whose purpose is to prevent excessive assist/hand-off suspension time. The SCSM sets the timer $T_{ASSIST/HAND\text{-}OFF}$ when the SCSM sends the establish temporary connection or select route/connect with a correlation ID operation. This timer is stopped when the SCSM receives the assist request instructions operation from the assisting/handed-off SSF. On expiration of $T_{ASSIST/HAND\text{-}OFF}$, the SCSM informs service logic and the maintenance functions, and the SCSM transits to the idle state.

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.

The latter operations can be sent to the SSF in a series of TCAP messages or in a component sequence, while the former can be sent only one at a time, separated by EDP-R messages received by the SCSM. The call-processing operations are as follows:

- Analyse information;
- Collect information;
- Connect;
- Connect to resource;
- Continue;
- Disconnect forward connection;
- Establish temporary connection;
- Initiate call attempt;
- Release call;
- Select facility; and
- Select route.

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

In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. The outputs are presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.

3.1.2.5.1 State 1: “Idle”

The following events are considered in this state:

- (e1) SL_Invocation: This is an internal event caused by the service logic’s need to start a call. The SCSM transmits the initiate call attempt operation to the SSF;
- (E2) Query_from_SSF: This is an external event, caused by a reception of one of the following operations:
  - Assist Request Instructions (for the Service Hand-off case); and
  - DP-specific request instructions[6].

[6] DP-specific request instructions are following (see 2): TAnswer, TDisconnect, TermAttemptAuthorized, TMidCall, TNoAnswer, AnalysedInformation, TCalledPartyBusy, CollectedInformation OAnswer, OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer, OriginationAttemptAuthorized, RouteSelectFailure.

Both events cause a transition to state 2, preparing SSF instructions.

- (E3) Notification_from_SSF: This is an external event caused by the reception of initial DP or DP specific request instructions operations that notifies the detection of TDP_N in the SSF. This event causes a transition back to the same state.

3.1.2.5.2 State 2: "Preparing SSF instructions"

In this state, the SCF determines how to further process.

The following events are considered in this state:

- (e4) Processing_Completed: This is an internal event. In this case, the SCF has completed the processing of the instructions to the SSF. This event causes a response to be sent to the SSF and a transition to state 1, Idle;
- (e5) SR_Facilities_Needed: This is an (internal) event caused by the 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 3, Routing to Resource;
- (e6) Processing_Failure: This (internal) event causes an appropriate exception processing[7] and a transition back to state 1, Idle.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in Figure 19.

3.1.2.5.2.1 State 2.1: "Preparing SSF instructions"

The state 2.1, Preparing SSF instructions, is where the initial decision is made on whether the SDF information or a specialized resource is needed, whether queueing is supported, etc. In addition, the EDP-R-related processing is also performed in this state.

On entering this state, the SCSM starts or resets the timer $T_{SCF\text{-}SSF}$.

The following events are considered in this state:

- (e2.1) Non-Call_Processing_Instructions: This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one or more of the following operations to be issued to the SSF:
  - Apply charging;
  - Call information request;
  - Cancel (for call information request);
  - Furnish charging information;
  - Request report BCSM event;
  - Request notification charging event;
  - Reset timer; and
  - Send charging information.

This event causes a transition back to state 2.1, Preparing SSF instructions.

- (e2.2) SR_Facilities_Needed: This is an internal event, caused by the service logic when there is a need to use the SRF. This event maps into the SCSM event (e5).
- (e2.3) Call_Processing_Instruction_Ready (monitoring[8] not required): This is an internal event caused by the service logic when the final call-processing-related operation is ready and there is no armed EDP and no outstanding call information report, event notification charging or apply charging report operations. It causes one of the following operations to be issued to the SSF:
  - Analyse information;
  - Collect information;

[7] Here and further in this Recommendation, the exception processing is not defined. It is assumed, however, that it must include releasing all the involved resources and sending an appropriate response message to the SSF.

[8] Including call information report, notification charging event and/or the apply charging report when the "send calculation to SCP indication" parameter for apply charging operation is set to True.

- Connect;
- Continue;
- Release call;
- Select facility; and
- Select route.

This event maps into the SCSM event (e4).

- (e2.4) Call_Processing_Instruction_Ready (monitoring[9] required): This is an internal event caused by the service logic when a call-processing-related operation is ready and the monitoring of the call is required (e.g. an EDP is set, or there is an outstanding call information report, event notification charging or apply charging report, or there is a need to issue such a request). It causes one of the following operations to be issued to the SSF:
  - Analyse information;
  - Collect information;
  - Connect;
  - Continue;
  - Release call;
  - Select facility; and
  - Select route.

In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:

- Apply charging;
- Call information request;
- Cancel (for call information request);
- Furnish charging information;
- Request report BCSM event;
- Request notification charging event; and
- Send charging information.

This event causes a transition into state 2.3, Waiting for notification or request.

- (e2.5) Ready_for_Queueing_Processing: This is an internal event caused by the service logic when queueing of the call is required. This event causes a transition into State 2.2, Queueing FSM.
- (e2.6) Processing_Failure: This is an internal event, and it maps into the SCSM event (e6) Processing_Failure.

3.1.2.5.2.2 State 2.2 "Queueing FSM"

When the SCF is processing the Query from the SSF/CCF, it may find that the resource to which the call must be routed is unavailable. One possible reason causing the resource to be unavailable is the "busy" condition[10]. 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:

- Hold call in network;
- Apply charging;
- Call information request;
- Cancel (for call information request);

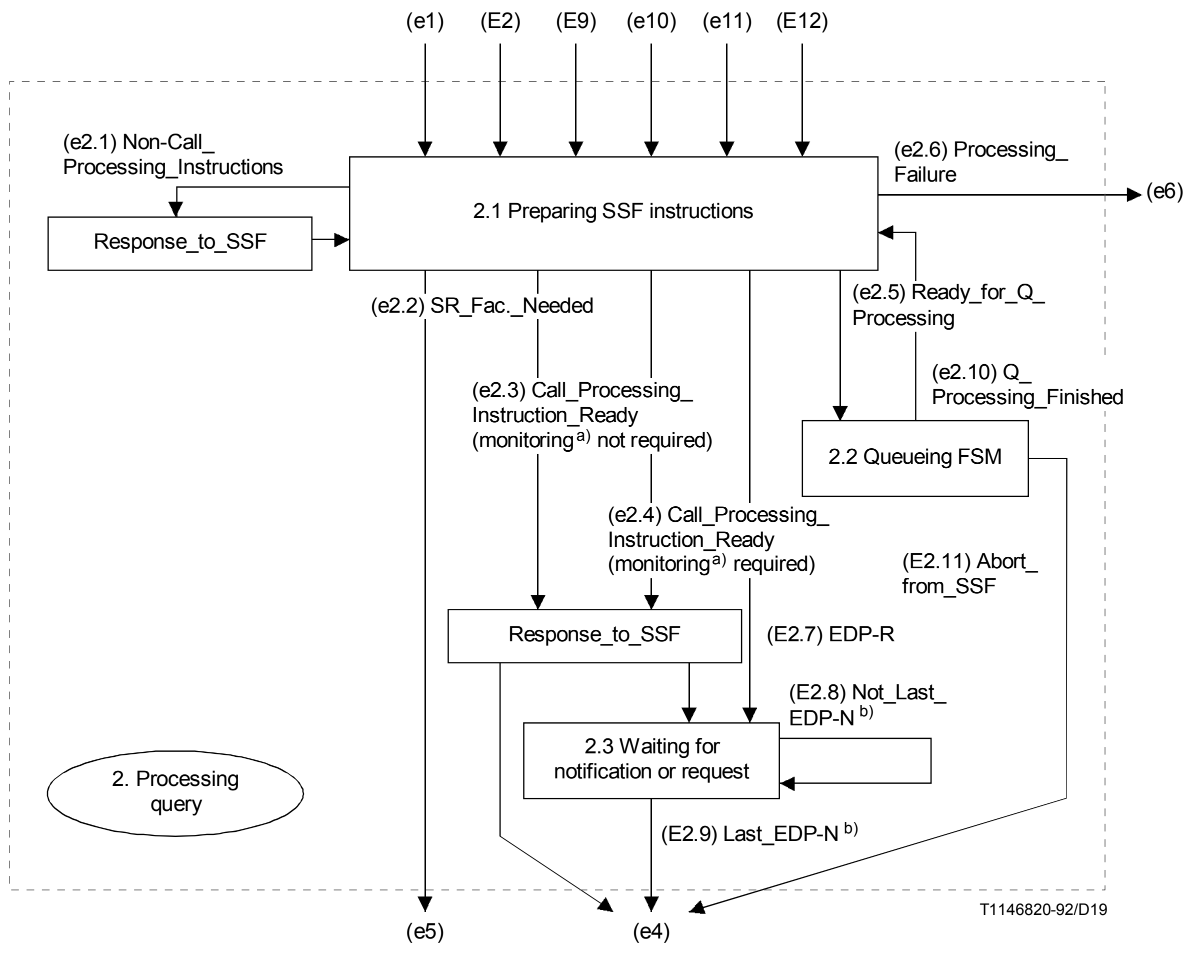
[9] Including call information report, notification charging event and/or the apply charging report when the "send calculation to SCP indication" parameter for apply charging operation is set to True.

[10] The manner in which the status of the resources is maintained is described in 3.1.2.4.5.

- Furnish charging information;
- Request report BCSM event;
- Request notification charging event;
- Reset timer; and
- Send charging information.

The following events are considered in this state:

- (e2.10) Queueing_Processing_Finished: This is an internal event caused by the service logic program when it is ready to prepare the call-related operation to the SSF. This event causes a transition to state 2.1, Preparing SSF instructions.
- (E2.11) Abort_from_SSF: This is an external event caused by the reception of the Abort message from the SSF (on call abandonment), and it causes a transition that maps into the SCSM event (e4).


(e1)
(E2)
(E9)
(e10)
(e11)
(E12)
(e2.1) Non-Call_ Processing_Instructions
Response_to_SSF
2.1 Preparing SSF instructions
(e2.6) Processing_ Failure
(e6)
(e2.2) SR_Fac._Needed
(e2.5) Ready_for_Q_ Processing
(e2.10) Q_ Processing_Finished
(e2.3) Call_Processing_ Instruction_Ready (monitoring a) not required)
2.2 Queueing FSM
(e2.4) Call_Processing_ Instruction_Ready (monitoring a) required)
(E2.11) Abort_ from_SSF
Response_to_SSF
(E2.7) EDP-R
(E2.8) Not_Last_ EDP-N b)
2.3 Waiting for notification or request
2. Processing query
(E2.9) Last_EDP-N b)
(e5)
(e4)
T1146820-92/D19

[a] Including call information request, apply charging with report request and request notification charging event.

[b] Including call information report, apply charging report and event notification charging.

FIGURE 19/Q.1218

Partial expansion of the state 2.FSM

This state further expands into an FSM, which is depicted in Figure 20.

This FSM does not explicitly describe all possible combinations of resource monitoring functions used for queueing. The following possibilities may be used in implementations:

- Request first status match report by means of the SCME;
- Request current status change report by means of the SCME;
- Request every status change report by means of the SCME; and
- Monitoring based on issuing by the SCSM of the request event report BCSM operation and subsequent reception of the event report BCSM operation to report the availability of the resource. Both the request and report occur in a single different call context. In this case, the Query and Update operations to the SDF or equivalent SCF functionality may be used for scanning the status of resources.

In the rest of this subclause, the state-by-state description of the FSM is followed by the description of the supporting mechanisms of the SCME.

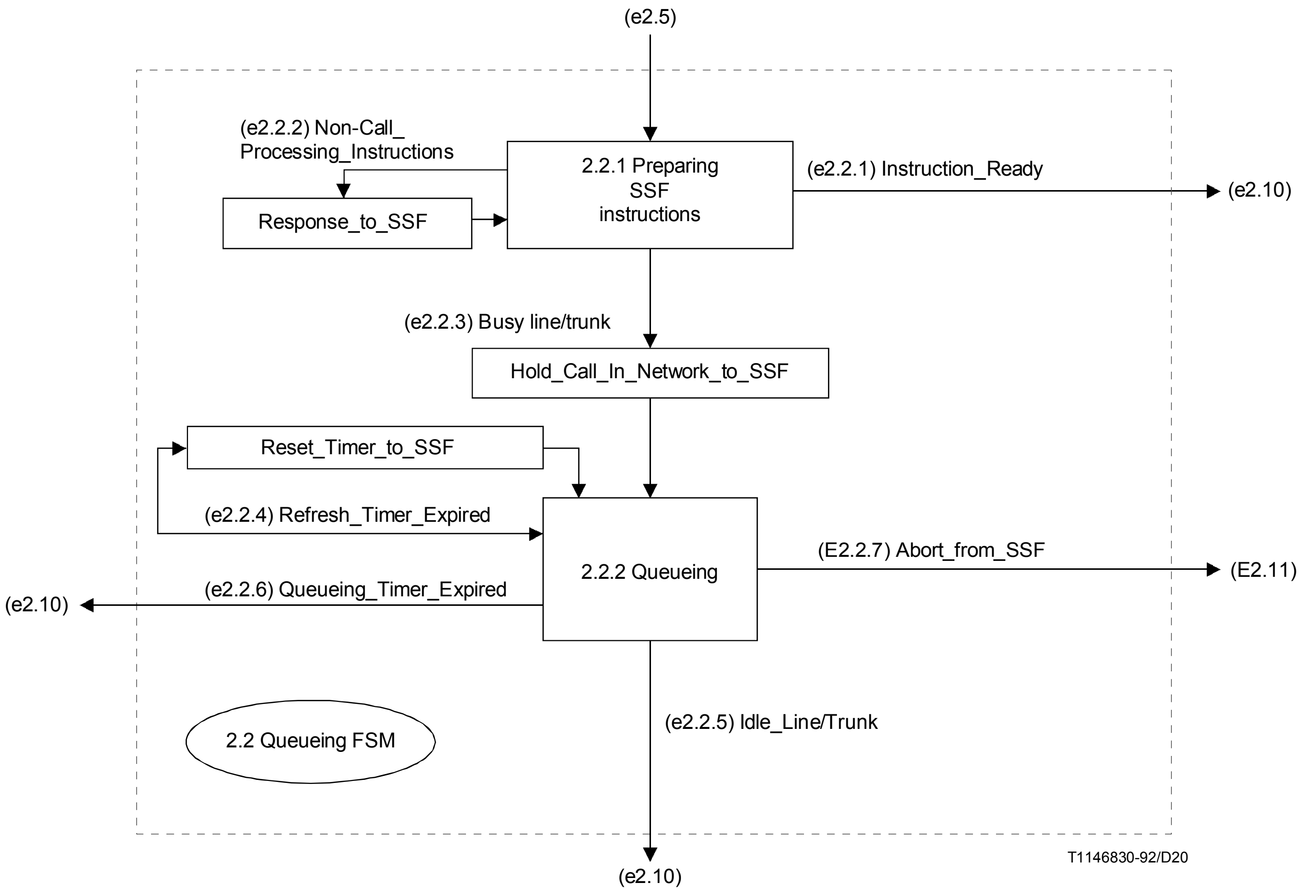

(e2.5)
(e2.2.2) Non-Call_ Processing_Instructions
2.2.1 Preparing SSF instructions
(e2.2.1) Instruction_Ready
(e2.10)
Response_to_SSF
(e2.2.3) Busy line/trunk
Hold_Call_In_Network_to_SSF
Reset_Timer_to_SSF
(e2.2.4) Refresh_Timer_Expired
(E2.2.7) Abort_from_SSF
2.2.2 Queueing
(E2.11)
(e2.2.6) Queueing_Timer_Expired
(e2.10)
(e2.2.5) Idle_Line/Trunk
2.2 Queueing FSM
T1146830-92/D20
(e2.10)

FIGURE 20/Q.1218

Partial expansion of the state 2 FSM as related to queueing

3.1.2.5.2.2.1 State 2.2.1: Preparing SSF instructions

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

- (e2.2.1) Instruction_Ready: This is an internal event that takes place only when the required resource is available. In this case, the SCSM has obtained the address of the free resource via the Get_Resource method of the Resource Control Object (see section 3.1.2.4.5). This event causes a transition to the state 2.1 Preparing SSF instructions [transition (e2.10)].

– (e2.2.2) Non-Call_Processing_Instructions: This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one or more of the following operations to be issued to the SSF:

a) Apply charging;

b) Call information request;

c) Cancel (for call information request);

d) Furnish charging information;

e) Request report BCSM event;

f) Request notification charging event;

g) Reset timer; and

h) Send charging information.

This event causes a transition back to state 2.2.1, Preparing SSF instructions.

– (e2.2.3) Busy_Line/Trunk: This is an internal event caused by the resource control object when no terminating line/trunk is available. This event causes the hold call in network operation to be sent to the SSF, and a transition to state 2.2.2, Queueing.

3.1.2.5.2.2.2 State 2.2.2: "Queueing"

In this state, the SCSM is awaiting an indication from the resource control object to proceed with routing a call to an idle trunk/line. The support of playing various announcements is also provided when the SCSM is in this state. In this Recommendation, the relevant further expansion of the state is not provided; it is, however, not different from that of the SCSM States 3 and 4. Once the SCSM enters this state, the queueing timer is started, and the $T_{SCF\text{-}SSF}$ is reset. The respective roles of these timers are as follows:

1) The queueing timer limits the time that a call can spend in the queue, and its value may be customer-specific.

2) The $T_{SCF\text{-}SSF}$ signals when the reset timer operation must be sent to the SSF/CCF lest the latter abandons the call. Hence, the value of this timer is set in agreement with that of the relevant timer T_{SSF} within the SSF/CCF.

The following events are considered in this state:

– (e2.2.4) Refresh_Timer_Expired: This is an internal event, which results in sending the reset timer operation to the SSF/CCF and a transition back to the same state.

– (e2.2.5) Idle_Line/Trunk: This is an internal event, which maps into the state 2 event (e2.10).

– (e2.2.6) Queueing_Timer_Expired: This is an internal event, which results in processing the cancel method of the resource control object and causes a transition out of this state to the state 2.1 Preparing SSF Instructions [Transition (e2.10)] (following procedures depends on the service logic's decision that may play (or not play) the terminating announcement).

– (E2.2.7) Abort_from_SSF: This is an external event caused by the reception of the Abort message from the SSF (on call abandonment), and it causes a transition that maps into the state 2 event (E2.11). The service control managing entity (SCME) takes care of updating the queueing data via the cancel method of the resource control object.

3.1.2.5.2.3 State 2.3: "Waiting for notification request"

In this state, the SCSM waits for a notification or a request from the SSF.

On entering this state the SCSM stops the timer $T_{SCF\text{-}SSF}$.

The following events are considered in this state:

– (E2.7) EDP-R: This is an external event, caused by a reception of one of the following operations:

a) Event report BCSM (for EDP_R); and

b) DP-specific request instructions family of operations.

This event causes a transition to state 2.1 Preparing SSF instructions.

- (E2.8) Not_Last_EDP-N: This is an external event, caused by a reception of one of the following operations:
  a) Apply charging report;
  b) Call information report;
  c) Event report BCSM (for EDP_N);
  d) Event notification charging; and
  e) DP-specific request instructions family of operations .

In this case, there is still an outstanding armed EDP[11]. This event causes a transition to back to state 2.3 Waiting for notification or request.

- (E2.9) Last_EDP-N: This is an external event, caused by a reception of one of the following operations:
  1) Apply charging report;
  2) Call information report;
  3) Event report BCSM (for EDP-N);
  4) Event notification charging;
  5) DP-specific request instructions family operations.

In this case, there is no outstanding armed EDP[11]. This event maps into the SCSM event (e4).

This concludes the description of state 2 Preparing SSF instructions.

3.1.2.5.3 State 3: "Routing to resource"

The resource is any SRF facility (e.g. intelligent peripheral).

In this state, interactions with the SSF are necessary. Accordingly, the following events cause transitions out of this state:

- (e7) Resource_Attached: The SRF is available. This event causes a transition to state 4, User interaction;
- (e8) Hand-off_Needed: When the hand-off procedure is initiated, the SCSM terminates the interaction with the initiating SSF. This event causes a transition to the state 1, idle. When the Assist request instructions operation from the handed-off SSF is received by the SCME, the SCME creates the new SCSM;
- (E9) Failure_from_SSF: The inability of the SSF to connect to requested resources causes a transition to state 2, Preparing SSF instructions; and
- (e10) Timer_Expired: This event takes place when $T_{ASSIST/HAND\text{-}OFF}$ expires. This event causes a transition to state 2, Preparing SSF instructions.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in Figure 21.

3.1.2.5.3.1 State 3.1: "Determine mode"

In this state, the SCSM determines the user interaction mode to connect the call to SRF. The following events are considered in this state:

- (e3.1) Instruction_Ready: This is an internal event that takes place only when the Initiating SSF relay case. In this case, the SCSM sends the connect to resource operation accompanied by play announcement or prompt & collect user information operation to the Initiating SSF, and transits out to the state 4 User interaction. This transition maps into the event (e7);
- (e3.2) Assist_Needed: This event is an internal event that takes place when the Assisting SSF is needed or Direct SCF-SRF relation is needed. In this case, the SCSM sends the establish temporary connection operation to the Initiating SSF with the Assisting SSF address or SRF address, and transits to the state 3.2 Waiting for assist request instructions; and

[11] Including call information report, notification charging event and/or the apply charging report when the "Send calculation to SCF indication" parameter for apply charging operation is set to True.

– (e3.3) Hand-off_Needed: This is an internal event that takes place only with the hand-off case. In this case, the SCSM sends the connect or select route operation with the handed-off SSF address to the initiating SSF, and transits to the state 1 idle.

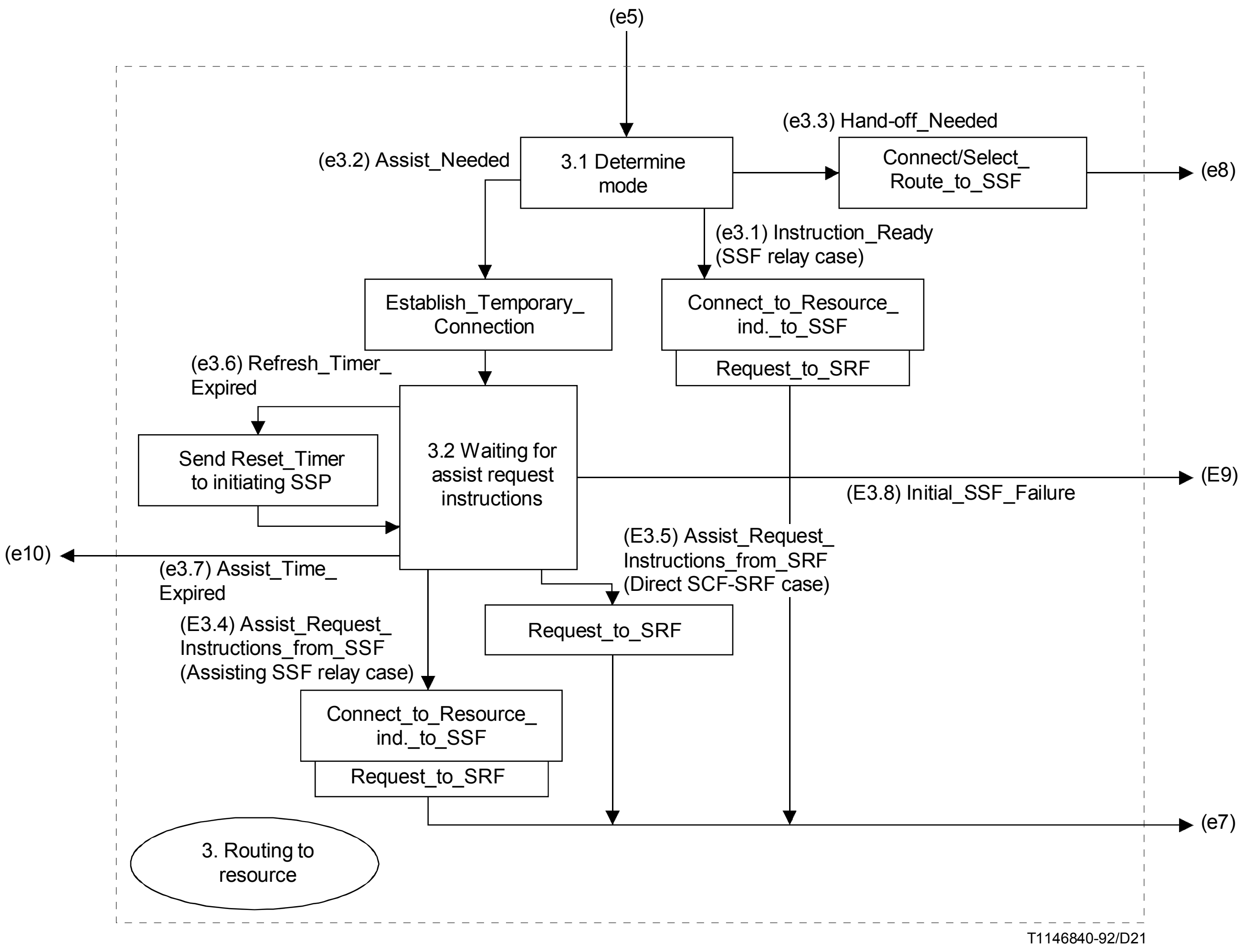

(e5)
(e3.3) Hand-off_Needed
(e3.2) Assist_Needed
3.1 Determine mode
Connect/Select_ Route_to_SSF
(e8)
(e3.1) Instruction_Ready (SSF relay case)
Establish_Temporary_ Connection
Connect_to_Resource_ ind._to_SSF
Request_to_SRF
(e3.6) Refresh_Timer_ Expired
Send Reset_Timer to initiating SSP
3.2 Waiting for assist request instructions
(E9)
(E3.8) Initial_SSF_Failure
(E3.5) Assist_Request_ Instructions_from_SRF (Direct SCF-SRF case)
(e10)
(e3.7) Assist_Time_ Expired
(E3.4) Assist_Request_ Instructions_from_SSF (Assisting SSF relay case)
Request_to_SRF
Connect_to_Resource_ ind._to_SSF
Request_to_SRF
(e7)
3. Routing to resource
T1146840-92/D21

FIGURE 21/Q.1218

The state 3 FSM

3.1.2.5.3.2 State 3.2: "Waiting for assist request instructions"

In this state, the SCSM waits for the assist request instructions operation from the Assisting SSF (SSF relay case) or from the SRF (Direct SCF-SRF case). On entering this state the SCSM starts the timer $T_{ASSIST/HAND-OFF}$, and resets the timer $T_{SCF-SSF}$. The following events are considered in this state:

– (E3.4) Assist_Request_Instructions_from_SSF (Assisting SSF relay case): This is an external event caused by the receipt of assist request instructions from the Assisting SSF. In this case, the SCSM transmits the connect to resource operation accompanied by play announcement or prompt & collect user information operation to the Assisting SSF, and transits out to the state 4, User interaction. This transition maps into the event (e7);

– (E3.5) Assist_Request_Instructions_from_SRF (Direct SCF-SRF case): This is an external event caused by the receipt of assist request instructions from the SRF. In this case, the SCSM transmits the play announcement or prompt & collect user information operation to the SRF, and transits out to the state 4, User interaction. This transition maps into the event (e7);

– (e3.6) Refresh_Timer_Expired: This is an internal event that takes place on the expiration of $T_{SCF-SSF}$. In this case, the SCSM transmits the reset timer operation to the Initiating SSF, and transits back to the same state;

- (e3.7) Assist_Timer_Expired: This is an internal event that takes place on the expiration of $T_{ASSIST/HAND-OFF}$. In this case, the SCSM informs the SCME and service logic program, and transits to the state 2 Preparing SSF instructions. This event maps into the event (e10); and
- (E3.8) Initial_SSF_Failure: This is an external event caused by the reception of SSF failure. This event causes a transition that maps into the SCSM event (E9).

3.1.2.5.4 State 4: "User interaction"

In this state, the SCF requests the SRF to provide user interaction (e.g. collect additional information and/or play announcements). When an interaction is finished the SCF can instruct the SSF to disconnect the bearer between SSF and SRF. Alternatively, it can send a user interaction operation to the SRF containing an indication that allows the SRF initial disconnection.

On entering this state the SCSM resets the timer $T_{SCF-SSF}$.

The following events cause transitions out of this state:

- (e11) Continue_SCF_Processing: In this case, the SCF has obtained all the information from the SRF, that is needed to instruct the SSF to complete the call. This event causes a transition to state 2, Preparing SSF instructions.
- (E12) Failure_from_SRF: In this case, the SCF has detected that
  a) The chosen resource cannot perform its function; and
  b) The chosen resource cannot be replaced.

Accordingly, this event causes a transition to state 2, Preparing SSF instructions.

To consider the processing of this state in more detail, it is expanded into a separate FSM depicted in Figure 22.

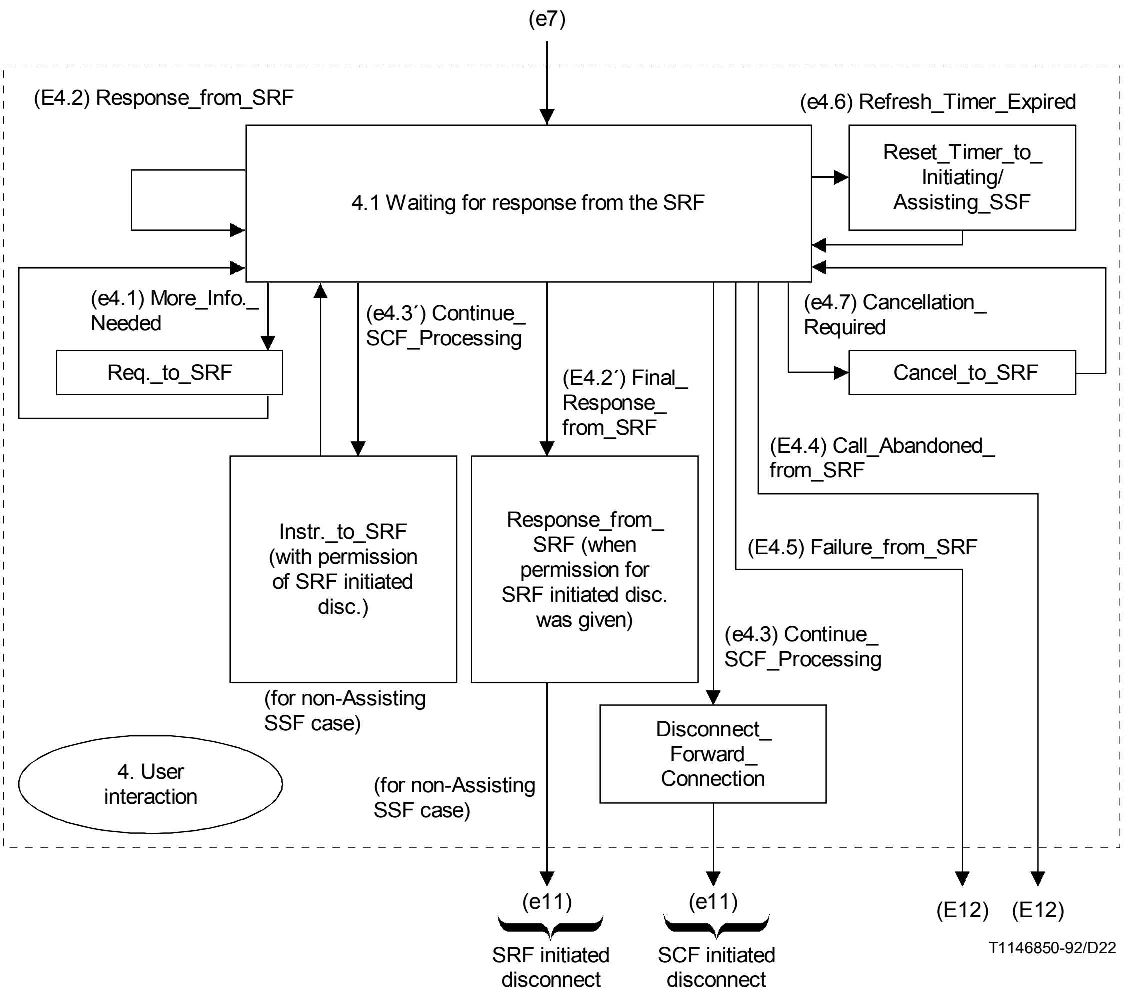
3.1.2.5.4.1 State 4.1 "Waiting for response from the SRF"

In this state, the SCF waits for the response to the previously sent operation and evaluates this response. The following events are considered in this state:

- (e4.1) More_Information_Needed results in issuing yet another operation to the SRF; it causes a transition back to State 4.1;
- (E4.2) Response_from_SRF: This is an external event caused by the reception of specialized resource report or return result from prompt and collect User information operation. On the receipt of this operation, the SCSM transits back to the same state;
- (E4.2′) Final_Response_from_SRF: This is an external event caused by the reception of specialized resource report operation in response to the previous play announcement or return result for prompt & collect user information operation with permission of SRF-initiated disconnect. In the Initiating SSF relay case and the Direct SCF-SRF case, on the receipt of this event, the SCSM transits to state 2, Preparing SSF instructions. This event maps into the event (e11);
- (e4.3) Continue_SCF_Processing: This is an internal event that takes place when the SCSM finishes the user interaction and requests the disconnection of bearer connection between the Initiating SSF and SRF by means of SCF initiated disconnect. In this case, the SCSM sends the disconnect forward connection operation to the Initiating SSF and transits to state 2, Preparing SSF instructions. This event maps into the event (e11);
- (e4.3′) Continue_SCF_Processing: This is an internal event that takes place when the SCSM finishes the user interaction and requests the disconnection of bearer connection between the Initiating SSF and SRF by means of SRF-initiated disconnect. In this case, the SCSM sends the play announcement or prompt and collect operation with permission of SRF-initiated disconnect to the SRF. In the case of Assisting SSF, the SRF-initiated disconnect cannot be used. In this case, the SCSM transits back to the same state;
- Both events, (E4.4) Call_Abandoned_from_SRF and (E4.5) Failure_from_SRF, are mapped into the event (E12) Failure_from_SRF, and the SCSM transits to state 2, Preparing SSF instructions;
- (e4.6) Refresh_Timer_Expired: This is an internal event that takes place on the expiration of $T_{SCF-SSF}$. In this case, the SCSM transmits the reset timer operation to the Initiating/Assisting SSF, and transits back to the same state; and

– (e4.7) Cancellation_Required: This is an internal event that takes place when the SCSM cancels the previous play announcement or prompt & collect user information operation. In this case, the SCSM sends the cancel operation to the Assisting SSF (SSF relay case) or the SRF (Direct SCF-SRF case), and transits back to the same state.

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



(e7)
(E4.2) Response_from_SRF
(e4.6) Refresh_Timer_Expired
4.1 Waiting for response from the SRF
Reset_Timer_to_ Initiating/ Assisting_SSF
(e4.1) More_Info._ Needed
Req._to_SRF
(e4.3´) Continue_ SCF_Processing
(E4.2´) Final_ Response_ from_SRF
(e4.7) Cancellation_ Required
Cancel_to_SRF
(E4.4) Call_Abandoned_ from_SRF
Instr._to_SRF (with permission of SRF initiated disc.)
Response_from_ SRF (when permission for SRF initiated disc. was given)
(E4.5) Failure_from_SRF
(e4.3) Continue_ SCF_Processing
(for non-Assisting SSF case)
Disconnect_ Forward_ Connection
4. User interaction
(for non-Assisting SSF case)
(e11)
SRF initiated disconnect
(e11)
SCF initiated disconnect
(E12)
(E12)
T1146850-92/D22

FIGURE 22/Q.1218

The state 4 FSM

3.1.2.5.5 SDF Related states

The interaction with the SDF is possible from any state of the SCF. In the following subclause, the SDF related states are specified.

3.1.2.5.5.1 State 5: "SDF Request Idle"

This state is a sub-state of any state representing the case before the request to the SDF is issued. The following event is considered in this state:

– (e13) Request_to_SDF: This is an internal event, caused by the service logic's need to either collect additional information from the SDF or to update the SDF. This event causes a transition to state 6, Waiting for SDF response.

3.1.2.5.5.2 State 6: "Waiting for SDF response"

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

- (E14) Response_from_SDF: This is an external event, caused by the reception of the response to the Query or Update operation previously issued to the SDF. This event causes a transition out of this state to state 5 SDF request idle; and
- (E15) SDF_Response: This is an external event, caused by the reception of the SDF Response to the Query or Update operation previously issued to the SDF. This event informs the SCF that the requested operation may be delayed, and does not cause a transition out of this state, and SCSM still remains the state 6 Waiting for SDF response.

It should be noted that the SCF may change the TCAP timer value for processing operation to an implementation-dependent larger value on the receipt of (E15).

3.1.3 SRF application entity procedures

3.1.3.1 General

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

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

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes 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.

3.1.3.2 Model and interfaces

The functional model of the AE-SRF is shown in Figure 23; the ASEs interface to TCAP (to communicate with the SCF) as well as interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 23.

The interfaces shown in Figure 23 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-Primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of intelligent network application protocol (INAP) are defined in 2.

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

The primitive interface between the SRF FSM and the maintenance functions is an internal interface and is not subject for standardization in CS-1.

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

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

The management functions related to the execution of operation received from the SCF are executed by the SRF management entity (SRME). The SRME interfaces the different SRF call state models (SRSM) and the functional entity access manager (FEAM). Figure 24 shows the SRF FSM structure.

The model associates a finite state machine (FSM) with each initial interaction request from the SCF[12]. Thus, multiple initial requests may be executed concurrently and asynchronously by the SRF, which explains the need for a single

[12] Such a request is executed by the SCSM when it is in its state 4.

entity that performs the tasks of creation, invocation, and maintenance of the SRSM objects. This entity is called the SRF managing 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;

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

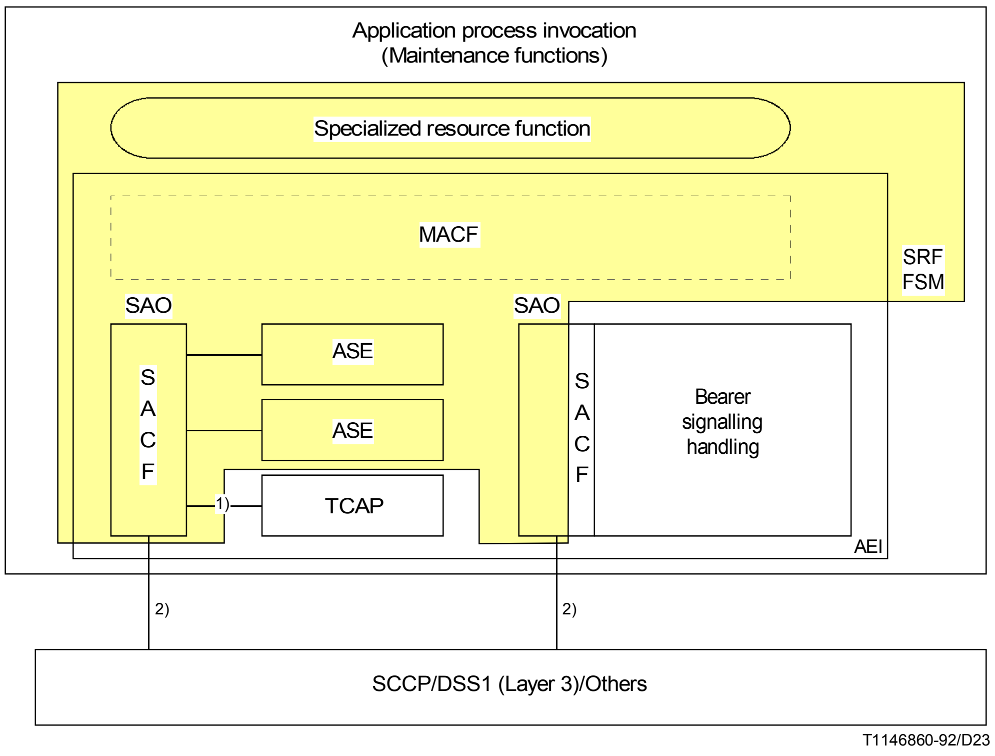
3) Handles the activity test functionality for the SCF-SRF relationship.

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.


Application process invocation
(Maintenance functions)
Specialized resource function
MACF
SRF
FSM
SAO
SAO
SACF
ASE
ASE
TCAP
1)
SACF
Bearer signalling handling
AEI
2)
2)
SCCP/DSS1 (Layer 3)/Others
T1146860-92/D23

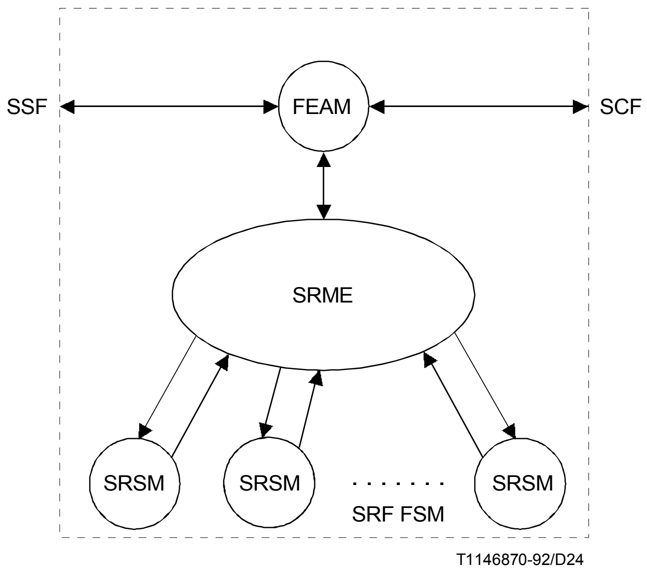
AEI	Application entity invocation
SRF	Specialized resource functions
FSM	Finite state machine
MACF	Multiple association control function
SACF	Single association control function
SAO	Single association object

[1)] TC-primitives or Q.932-primitives.
[2)] N-primitives.

NOTE – The SRF FSM includes several finite state machines.

FIGURE 23/Q.1218

Functional model of SRF AE



FEAM	Functional entity access manager
SRME	SRF management entity
SRSM	SRF call state model

FIGURE 24/Q.1218

SRF FSM structure

3.1.3.4 The SRSM

The SRSM is presented in Figure 25. In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. Finally, the outputs are presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.

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 play announcement or prompt & collect) 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 interaction is ABORTed.

In any state, if there is an error in a received operation, the maintenance functions are informed and the SRSM remains in the same state in which it received the erroneous operations; 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 or SSF is terminated, then the SRSM informs the SSF or SCF and returns to idle state after ensuring that all resources allocated to the call, including the dialogues with other co-located functions, have been de-allocated.

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 assist request instructions 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 specialized resource report operation or the return result for the prompt and collect user information 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.

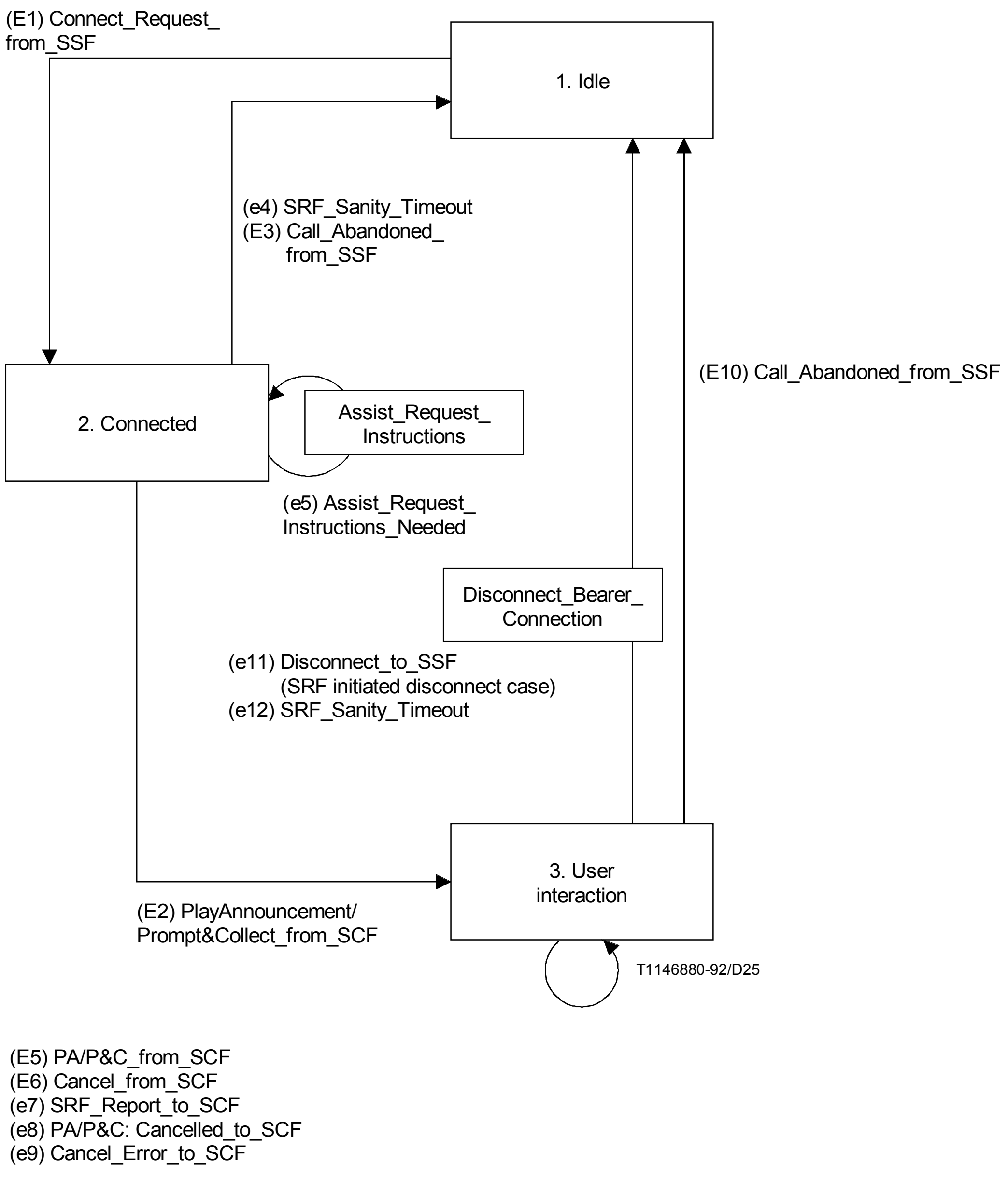

(E1) Connect_Request_ from_SSF
1. Idle
(e4) SRF_Sanity_Timeout
(E3) Call_Abandoned_ from_SSF
(E10) Call_Abandoned_from_SSF
2. Connected
Assist_Request_ Instructions
(e5) Assist_Request_ Instructions_Needed
Disconnect_Bearer_ Connection
(e11) Disconnect_to_SSF (SRF initiated disconnect case)
(e12) SRF_Sanity_Timeout
3. User interaction
(E2) PlayAnnouncement/ Prompt&Collect_from_SCF
T1146880-92/D25
(E5) PA/P&C_from_SCF
(E6) Cancel_from_SCF
(e7) SRF_Report_to_SCF
(e8) PA/P&C: Cancelled_to_SCF
(e9) Cancel_Error_to_SCF

FIGURE 25/Q.1218

The SRSM

3.1.3.4.1 State 1: "Idle"

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

- (E1) Connect_request_from_SSF: This event corresponds to a bearer signalling connection request message from the SSF. The details of the bearer signalling state machine related to establishing the connection are not of interest to the FSM. The SRSM goes to state "connected";
- (E3) Call_Abandoned_from_SSF: This event takes place when the SRSM receives a release message from the SSF in connected state, indicating that the call party has disconnected. 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 PA/P&C 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) Call_Abandoned_from_SSF: This event takes place when the SRSM receives a release message from the SSF in user interaction state, indicating that the call party has disconnected. The SRSM goes to state "idle";
- (e11) Disconnect_to_SSF: This event occurs when the SCF has enabled SRF initiated disconnect by the last PA/P&C from SCF (E2) or (E5) with the parameter. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending last specialized resource report operation to the SCF (e7). The SRSM goes to state "idle"; and
- (e12) SRF_Sanity_Timeout: This event occurs when the SRSM has been in user interaction state for a network-operator-defined period of time (timer T_{SRF}) without having a PA/P&C operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "idle".

3.1.3.4.2 State 2: "Connected"

This state represents the condition of the SRSM when a bearer channel has been established between a user and the SRF but the initial PA/P&C has not yet been received (e.g. when establish temporary connection 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) PA/P&C_from_SCF: This event takes place when the first play announcement or prompt & collect user information operation(s) from the SCF is received. The SRSM goes to state "user interaction".
- (E3) Call_Abandoned_from_SSF: This event takes place when the SRF receives a release message from the SSF, indicating that the call party has disconnected. 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 PA/P&C 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 assist request instructions operation is sent from the SRSM to the SCF in the absence of a PA/P&C event (E2) initiated by the presence of a PA/P&C operation concatenated with the setup request from SSF (E1) (Direct SCF-SRF case). No state change occurs as a result of this event.

3.1.3.4.3 State 3: "User interaction"

The user interaction state indicates that communication is occurring between the user and the SRF via the bearer channel established at the connected state. This state is entered as a result of event E2. It is exited as a result of events E10, e11 and e12. Events E5, E6, e7, e8 and e9 do not cause a state change. Event E5 also represents additional PA/P&C operations which are buffered as discussed in the procedures.

- (E2) and (E5) PA/P&C_from_SCF: This event takes place when an initial or subsequent play announcement or prompt & collect user information 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 PA/P&C): This event takes place when the corresponding play announcement or prompt & collect user information 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 specialized resource report operation is sent to the SCF. The SRSM remains in state "user interaction".

- (e8) PA/P&C_Cancelled_to_SCF: This event takes place when the PA/P&C error caused by the cancel (for play announcement or prompt & collect user information) operation is sent to the SRF. This event represents the successful cancellation of an active or buffered PA/P&C operation. The SRSM remains in state "user interaction".

- (e9) Cancel_Error_to_SCF: This event takes place when the cancel error (for play announcement or prompt & collect user information) is sent to the SRF. This event represents the unsuccessful cancellation of a PA/P&C operation. The SRSM remains in state "user interaction".

- (E10) Call_disconnect_from_SSF: This event takes place when the SRSM receives a release message from the SSF, indicating that the call party has disconnected. The SRSM goes to state "idle".

- (e11) Disconnect_to_SSF: This event occurs when the SCF has enabled SRF initiated disconnect with the last PA/P&C 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 last specialized resource report operation to the SCF. The SRSM goes to state "idle".

- (e12) 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 PA/P&C operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "idle".

In addition to these explicitly marked transitions, failure of a user-SRF bearer connection will cause the SRSM to transit to Idle from any state. These transitions are not shown on Figure 25 for the purpose of visual clarity.

3.1.3.5 Examples SRF control procedures

This subclause provides a detailed description of the SRF procedures. Arrow diagrams are used for the description of the connect, interaction with the end user, and disconnect stages.

The SRF control procedures are based on various physical allocation patterns of SRF. The various control procedures are described in this subclause in accordance with the example physical scenarios of protocol architecture in 0.2.

The service assist and 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.

3.1.3.5.1 SRF connect procedures

3.1.3.5.1.1 SRF connect physical procedures

Several procedures are required for different physical scenarios. The cases to be covered are described below and illustrated in Figure 26.

i) the IP is integrated into the SSP, or directly attached to the SSP, that is interacting with the SCP but the SCP's operations to the IP are relayed via the SSP which performs any needed protocol conversion;

ii) the IP is directly attached to the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved;

iii) the IP is integrated into another SSP, or directly attached to another SSP, than the one that is interacting with the SCP but the SCP's operations to the IP are relayed via the second SSP (called the "Assist" method), and on completion of the user interaction, control is returned to the first SSP;

iv) the IP is directly attached to another node 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 DSS1 using the facility IE (in this case, the SSP would have to do protocol conversion from SS No. 7 TCAP to DSS1 facility IE for the operations and responses it relayed between the SCP and the IP); the direct messaging between the SCP and the IP may be SS No. 7 TCAP based; and bearer control signalling may be any system.

Each of the scenarios will now be examined using arrow diagrams.

Case i) is illustrated in Figure 27. Note that for the integrated IP/SSP, the internal activities of the node can still be modelled in this way, but the details of how this is achieved are left to the implementor. This approach makes it unnecessary for the SCP to distinguish between integrated and external but directly connected IPs. See also a note on the possibility of concatenating the first user interaction operation with the connect to resource operation discussed in the subclause on user interaction below. The establishment of the SCF-SRF relationship in this case is implicit.

Case ii) requires that the IP indicate to the SCP that it is ready to receive operations (see Figure 28). The establishment of the SCF-SRF relationship is explicit. Note that it is necessary to convey a correlation ID to ensure that the transaction established between the SCP and the IP can be correlated to the bearer connection setup as a result of the SCP's preceding operation to the SSP.

Case iii) requires that a transaction be opened with the assisting SSP so that it may relay operations from the SCP to the IP (integrated or external). Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP. After the assist request instructions are received by the SCP, the procedures are the same as case i). Figure 29 illustrates the preamble involved.

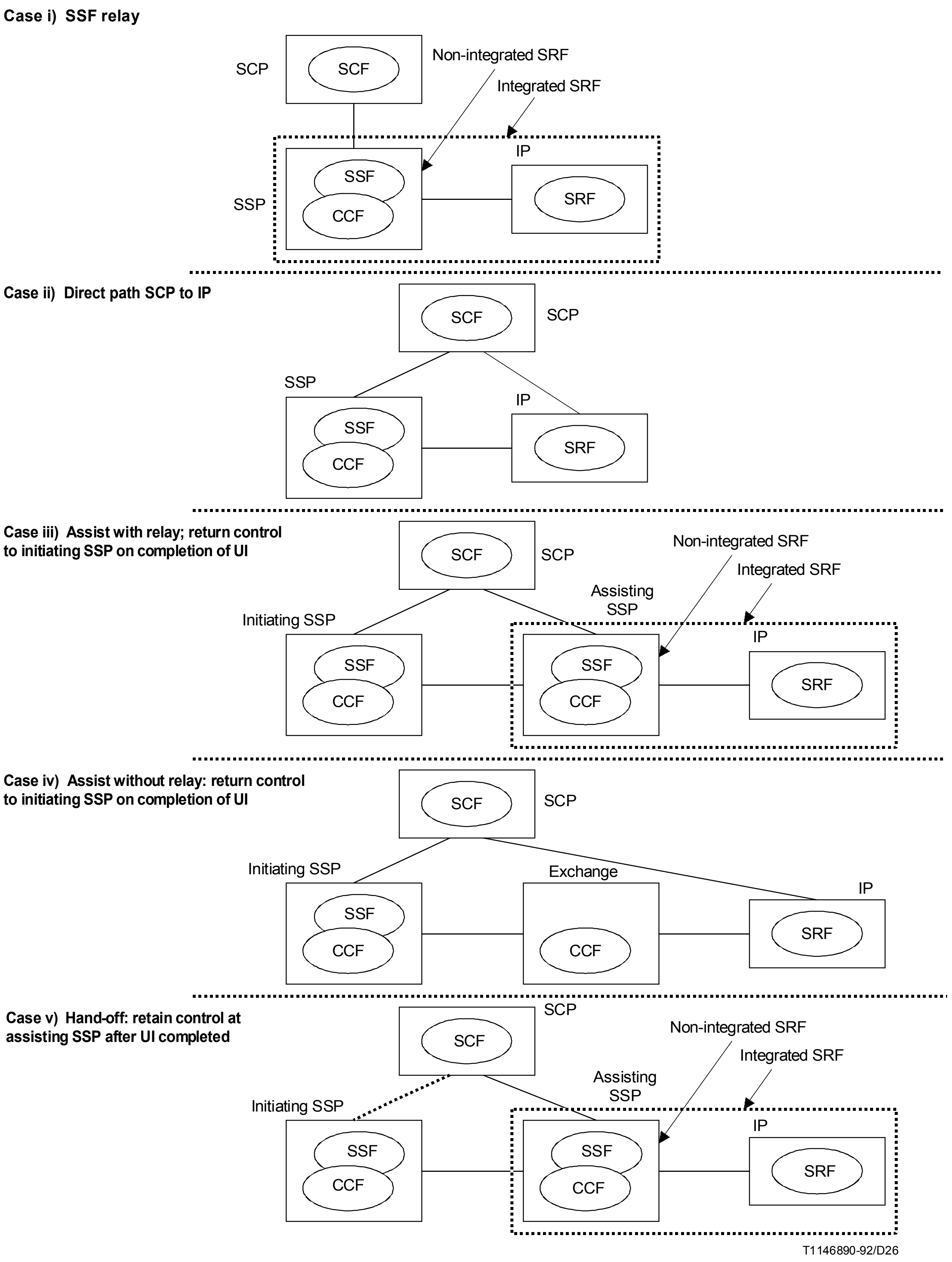


FIGURE 26/Q.1218

Physical scenarios

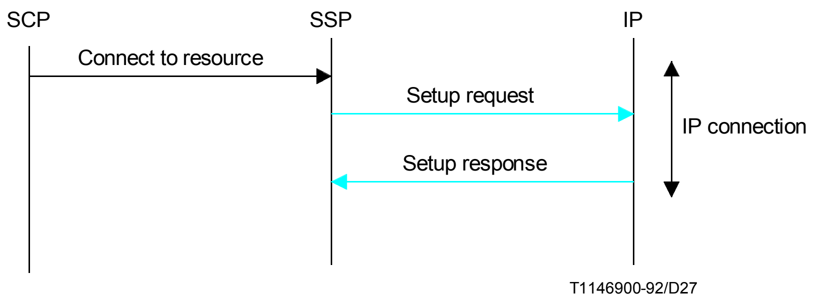


FIGURE 27/Q.1218

Connection to integrated or external IP with SSP relay of IP operations

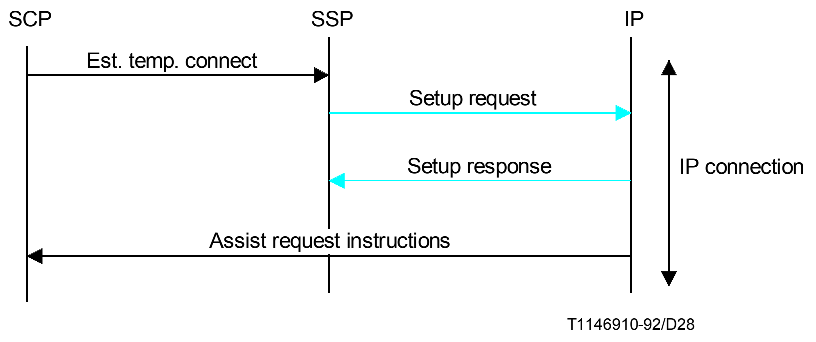


FIGURE 28/Q.1218

Connection to IP with direct link to SCP, IP initiates interaction with SCP

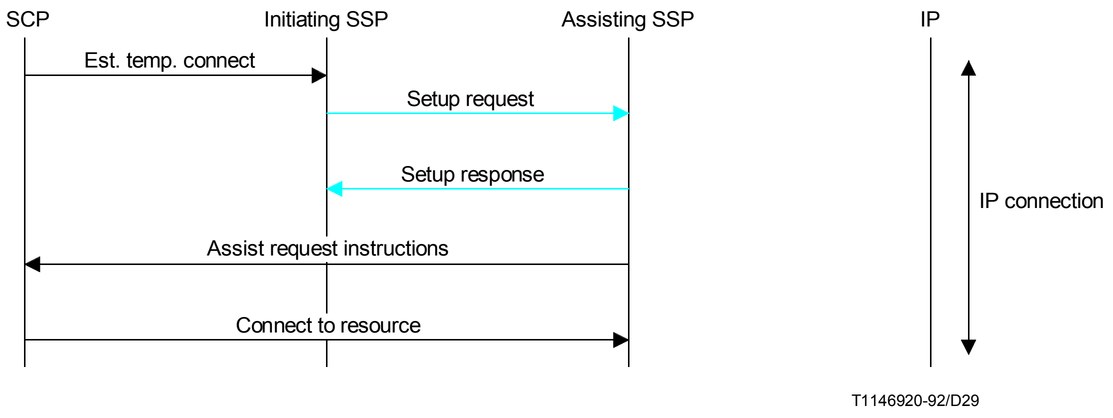


FIGURE 29/Q.1218

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 28 as shown in Figure 30.

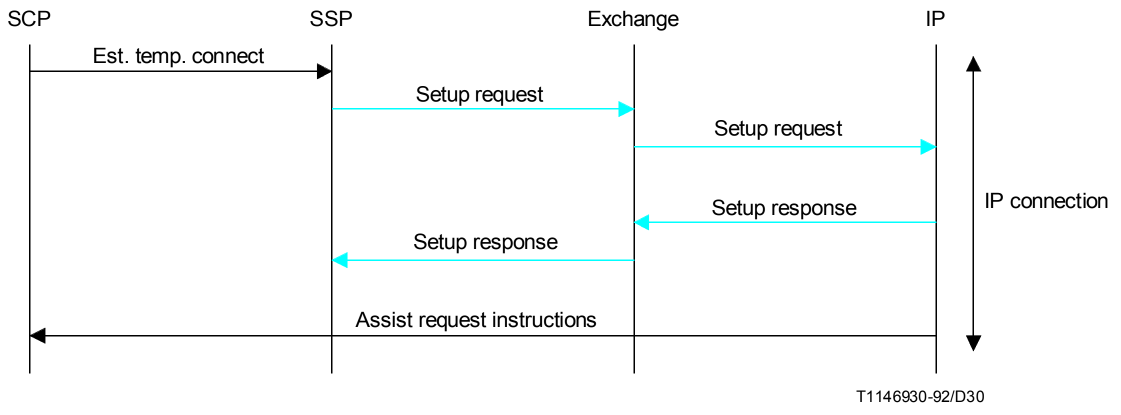


FIGURE 30/Q.1218

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 27 applies at handed-off SSP. This is shown in Figure 31. Note that the activity at handed-off SSP represents a new interaction with the SCP and "Assist Request Instructions" 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.

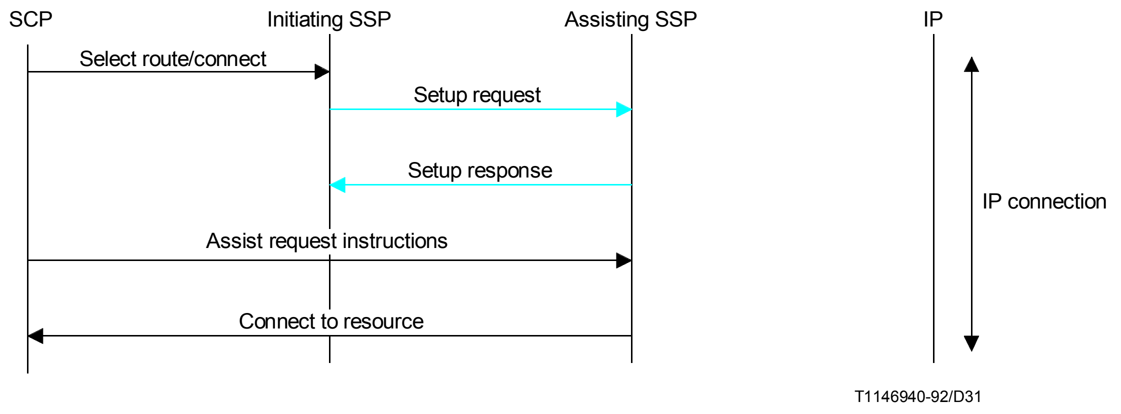


FIGURE 31/Q.1218

Preamble for hand-off case

3.1.3.5.2 SRF end user interaction procedures

The end user interaction procedures allow:

- the sending of one or multiple messages to the end user by using the play announcement operations;
- a dialogue with the end user by using one or a sequence of prompt & collect user information operations;
- a combination of the above; and
- cancellation of a play announcement or prompt & collect user information operations by using a generic cancel operation.

3.1.3.5.2.1 Play announcement/prompt & collect (PA/P&C)

There are only two physical scenarios for user interaction:

i) the SSP relays the operations from the SCP to the IP and the responses from the IP to the SCP (SSF relay case); and

ii) The operations from the SCP to the IP and the responses from the IP are sent directly between the SCP and the IP without involving the SSP (direct SCF-SRF case).

Case i) is illustrated in Figure 32 below.

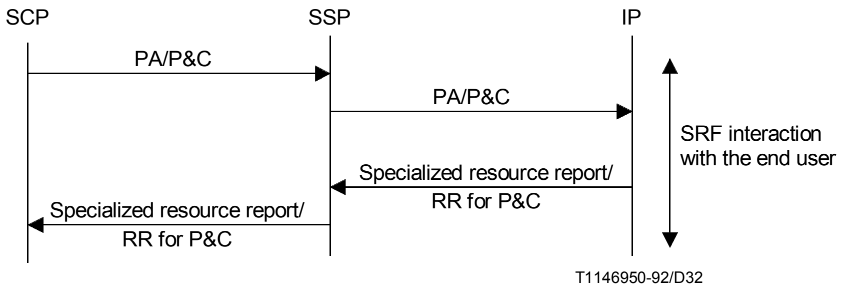

SCP
SSP
IP
PA/P&C
PA/P&C
SRF interaction with the end user
Specialized resource report/ RR for P&C
Specialized resource report/ RR for P&C
T1146950-92/D32

FIGURE 32/Q.1218

SSP relay of user interaction operations and responses

Case ii) is illustrated in Figure 33 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 27, the connect to resource and the first PA/P&C 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.

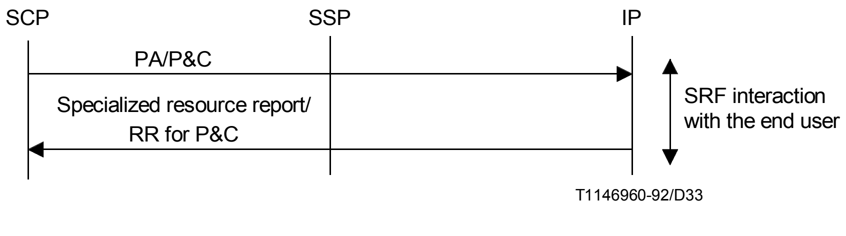

SCP
SSP
IP
PA/P&C
Specialized resource report/ RR for P&C
SRF interaction with the end user
T1146960-92/D33

FIGURE 33/Q.1218

Direct SCF-SRF of user interaction operations and responses

3.1.3.5.3 SRF disconnection procedures

The disconnection procedures are controlled by the SCF and the procedure used is selected based on the needs of the service being executed. The bearer disconnection procedure selected by the SCF is to either allow the SRF to disconnect on completion of user interaction, or to have the SCF explicitly order the SSF to disconnect.

SRF disconnect does not cause disconnection by the SSF/CCF back to the end user terminal unless the transaction with the SCF has been terminated, indicating the user interaction completed the call. The SSF/CCF recognizes that a connection to an SRF is involved because the operations from the SCF for this purpose are distinct from the operations that would be used to route the call towards a destination. There is no impact on bearer signalling state machines as a result of this since incoming and outgoing bearer signalling events are not simply transferred to each other, but rather are absorbed in call processing, and regenerated as needed by call processing. Therefore, to achieve the desired functionality, call processing need simply choose not to regenerate the disconnect in the backward direction. Figure 34 illustrates this concept.

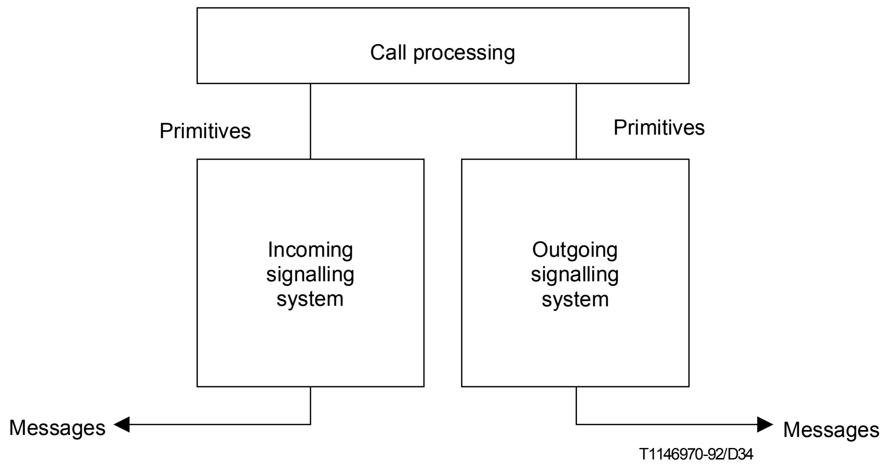


FIGURE 34/Q.1218

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 PA/P&C 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 PA/P&C. When a subsequent PA/P&C is received, it is buffered until the completion of any preceding PA/P&C.

2) A generic cancel operation terminates the indicated PA/P&C if it is being executed by the SRF, but does not disconnect the SRF. If the cancel operation is for a buffered PA/P&C, that PA/P&C is discarded, but the current and any buffered PA/P&Cs are executed. An SRF interacts with one user only and therefore cancelling a PA/P&C 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 PA/P&C. An SRF left connected without a PA/P&C to execute may autonomously disconnect if it has not received any PA/P&C operations within a defined time limit. This could occur, for example, after an establish temporary connection which is not followed within a reasonable time period with a PA/P&C 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 PA/P&C, then the SRF must disconnect on completion of the user interaction.

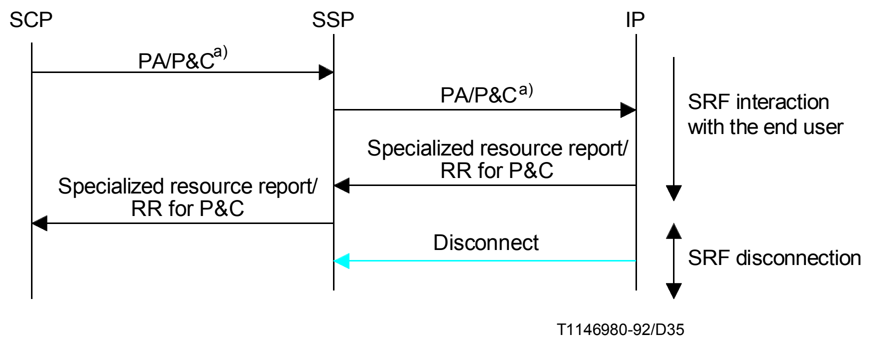
5) When SRF initiated disconnect is not enabled, the SCF must ask the SRF to inform it of the completion of the user interaction using the specialized resource report operation for "announcement complete" or using the return result for the prompt and collect user information operation.

6) If the user disconnects, the SRF is disconnected and the SSF releases resources and handles the transaction between the SSF and the SCF as specified in Recommendation Q.1214 and in this Recommendation. The SRF discards any buffered operations and returns its resources to idle. The relationship with the SCF is terminated.

7) When the SCF explicitly orders the SSF to disconnect by "disconnect forward connection" operation, the SSF releases the bearer connection to the SRF, and returns to the "waiting for instructions" state. No operation reporting SRF disconnect from the SSF to the SCF is required.

3.1.3.5.3.1 SRF initiated disconnect

The SRF disconnect procedure is illustrated in Figure 35. The SRF disconnect is enabled by the SCF within a PA/P&C operation. When the SRF receives a PA/P&C enabling disconnection, it completes the dialogue as instructed by the PA/P&C, 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 35 is the "handed-off" SSP.


SCP
SSP
IP
PA/P&C a)
PA/P&C a)
SRF interaction with the end user
Specialized resource report/ RR for P&C
Specialized resource report/ RR for P&C
Disconnect
SRF disconnection
T1146980-92/D35

a) Disconnect from SRF is not forbidden.

FIGURE 35/Q.1218

SCF disconnect for local, embedded and hand-off scenarios

For the assisting SSF case, the SRF initiated disconnect procedures are not used because the assisting SSF remains in the "waiting for instructions" state and does not propagate the disconnection of the bearer connection to the Initiating SSF. The SCF initiated disconnect procedures described in the following subclause are used for the assisting SSF case.

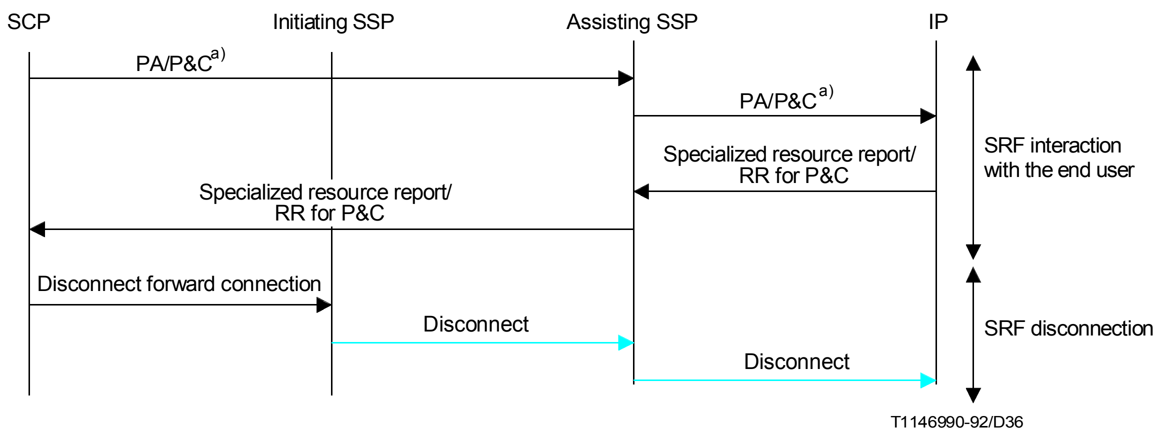
For the direct SCF-SRF case, the procedures also work in the same manner. The SRF disconnect is enabled by the SCF within a PA/P&C operation. When the SRF receives a PA/P&C enabling disconnection, it completes the dialogue as instructed by the PA/P&C, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The Initiating SSF/CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The Initiating SSF returns to the "waiting for instructions" state and executes any buffered operations.

3.1.3.5.3.2 SCF Initiated Disconnect

The SCF initiated disconnect procedure is illustrated in Figure 36. Bearer messages are shown in gray. The figure shows only the assisting SSR 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 PA/P&C operation requested. The specialized resource report operation contains an "announcement complete" and return result for P&C contains "collected information."

The SCF initiated disconnect uses an operation called disconnect forward connection. Once the disconnect forward connection operation 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 disconnect forward connection 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 establish temporary connection, does not disconnect back to the user but returns to the "waiting for instructions" state.


SCP
Initiating SSP
Assisting SSP
IP
PA/P&C a)
PA/P&C a)
Specialized resource report/ RR for P&C
Specialized resource report/ RR for P&C
SRF interaction with the end user
Disconnect forward connection
Disconnect
Disconnect
SRF disconnection
T1146990-92/D36

[a] Disconnect from SRF forbidden.

FIGURE 36/Q.1218

SCF initiated disconnect for assist scenario

3.1.3.5.4 Examples Illustrating Complete User Interaction Sequences

The following figures and their accompanying tables provide examples of complete sequences of user interaction operations covering the three stages:

– Connect the SRF and the end user (bearer connection) and establish the SCF-SRFrelationship.

– Interact with the end user.

– Disconnect the SRF and the end user (bearer connection) and terminate the SCF-SRF relationship.

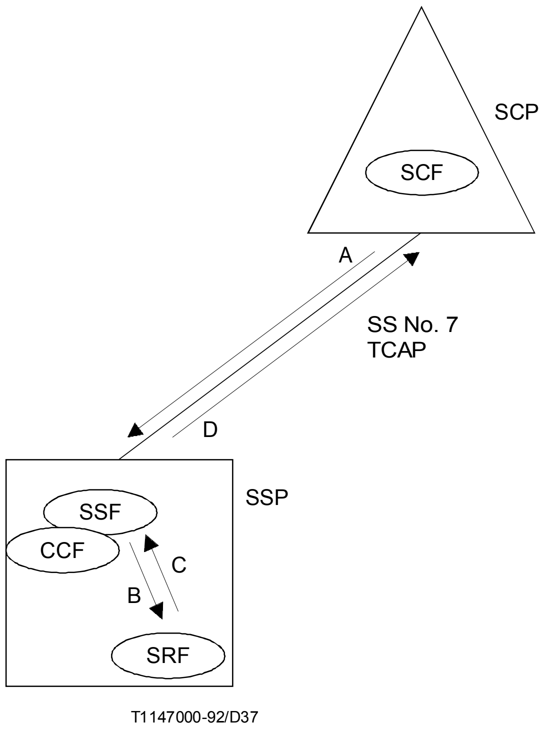


FIGURE 37/Q.1218

SSP with integrated SRF

In Figure 37, the SSP with an integrated (or embedded) SRF, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C	Connect to resource; PA/P&C Setup; PA/P&C	A B
User interaction	PA/P&C Specialized resource report/RR for P&C	A then B C then D
SRF initiated disconnect	Specialized resource report/RR for P&C Disconnect	C then D C (intra-SSP bearer control)
SCF initiated disconnect	Specialized resource report/RR for P&C Disconnect forward connection Disconnect	C then D A B (intra-SSP bearer control)

A simple extension to this integrated case is the configuration where the SRF is located in an intelligent peripheral locally attached to the SSP. The SCP-IP operations are relayed via the SSF in the SSP. This is depicted in Figure 38.

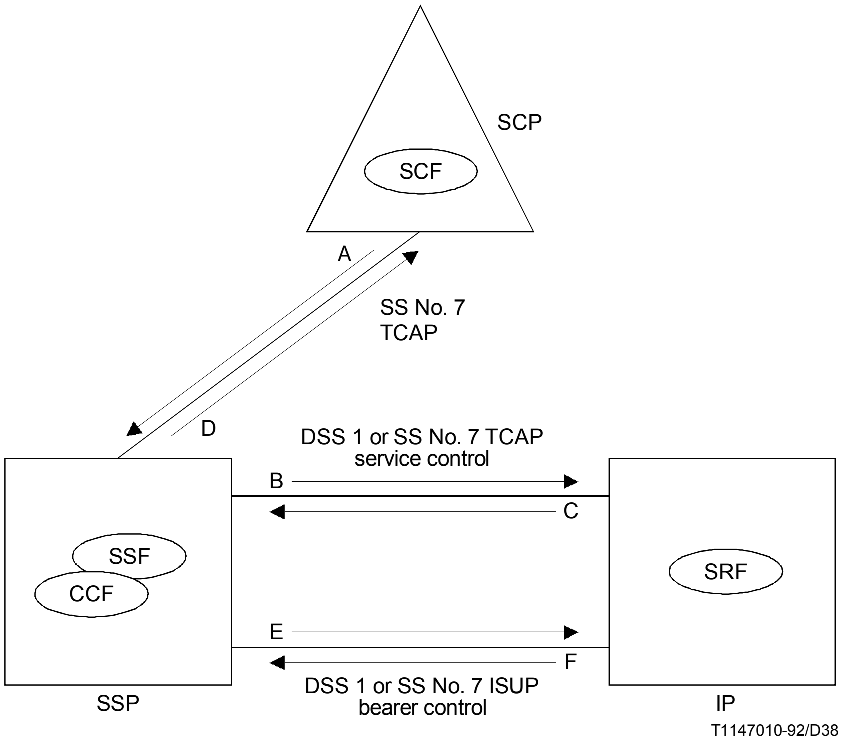


FIGURE 38/Q.1218

SSP relays messages between SCP and IP

The procedural scenarios for this relay SSF with an IP (Figure 38) can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C	Connect to resource; PA/P&C	A
	If DSS 1 used:	
	Setup; PA/P&C	E and B (Facility IE)
	If SS No. 7 used:	
	Setup	E
	PA/P&C	B
User interaction	PA/P&C	A then B
	Specialized resource report/RR for P&C	C then D
SRF initiated disconnect	Specialized resource report/RR for P&C	C then D
	Disconnect	F
SCF initiated disconnect	Specialized resource report/RR for P&C	C then D
	Disconnect forward connection	A
	Disconnect	E

In some cases, the IP may have an SS No. 7 or other interface to the controlling SCP. This case is shown in Figure 39. Note that the SCP must correlate two transactions to coordinate the activities.

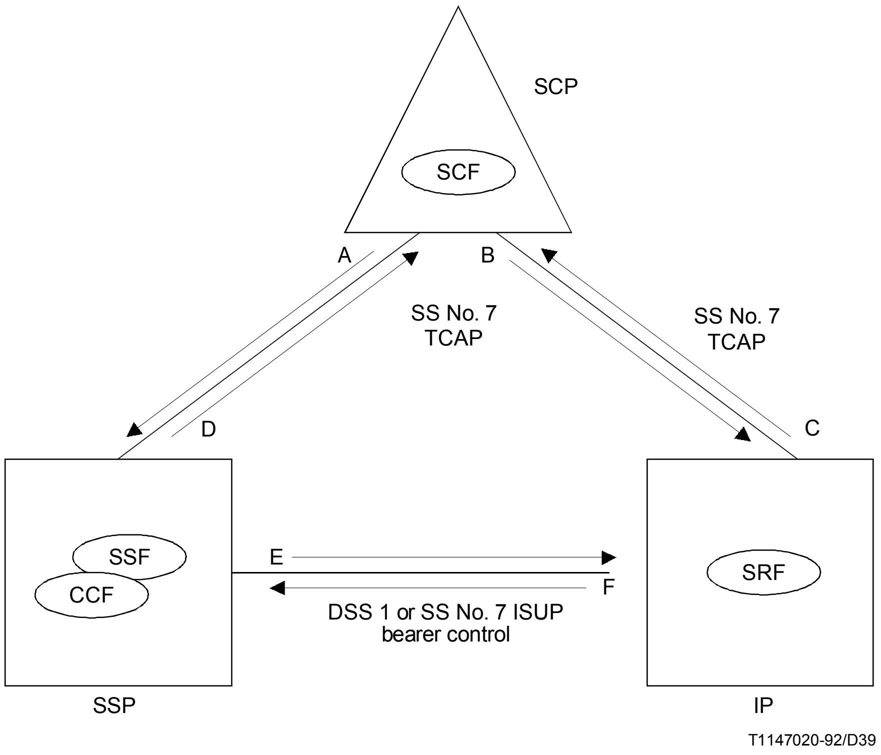


FIGURE 39/Q.1218

Direct SCP-IP information transfer

In Figure 39, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource	Est. temporary connection Setup Assist request instructions	A E C
User interaction	PA/P&C Specialized resource report/RR for P&C	B C
SRF initiated disconnect	Specialized resource report/RR for P&C Disconnect	C F
SCF initiated disconnect	Specialized resource report/RR for P&C Disconnect forward connection Disconnect	C A E

The assisting SSF scenario involves straightforward procedural extensions to the basic cases shown above. One mapping of the assisting SSF case is shown in Figure 40. In this case, SRF initiated disconnect cannot be used. Other physical mappings can be derived as described in the text following the figure and its accompanying table.

Note that the integrated SRF and SSF relay case requires a transaction between the SCP and the assisting SSP (Figure 40) but the SCP direct case does not since the transaction is directly between the SCP and the IP connected to the remote exchange. In the latter case, any transit exchanges, including the one the IP (SRF) is connected to, are transparent to the procedures.

Note also that the SCP must again correlate two transactions.

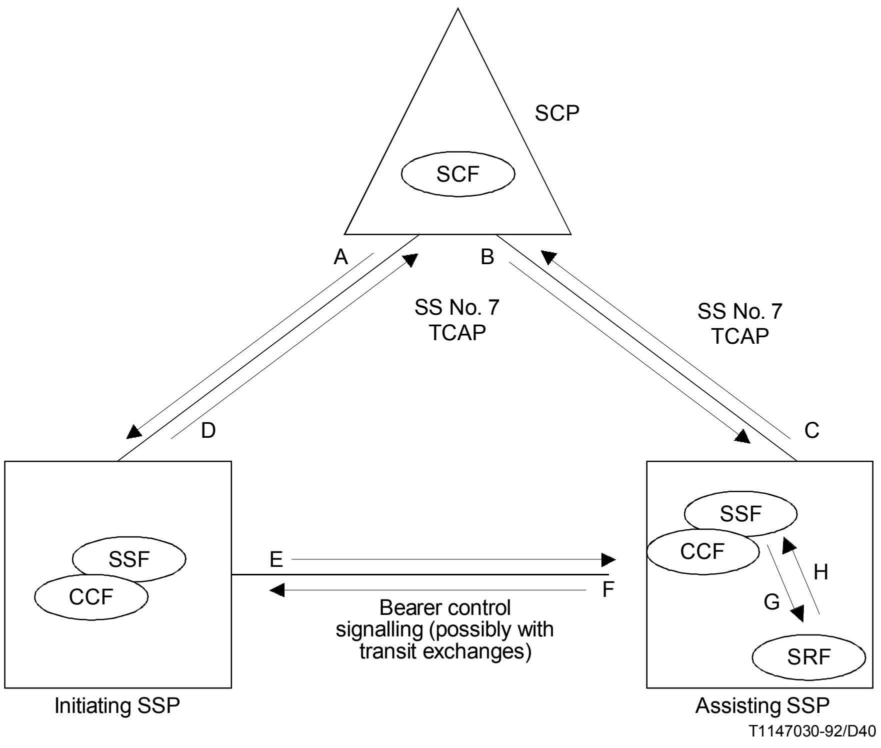


FIGURE 40/Q.1218

SSP assist (relay SSP)

In Figure 40, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Assist preamble	Establish temp. connection Setup Assist request instructions Connect to SR; PA/P&C Setup PA/P&C	A E C B G G
User interaction	PA/P&C Specialized resource report/RR for P&C	B then G H then C
SCF initiated disconnect	Specialized resource report/RR for P&C Disconnect forward connection Disconnect	H then C A F and G (intra-SSP bearer ctrl)

Note that the assisting SSP case shown in Figure 40 can be generalized to cover both the case where the SRF is embedded in assisting SSP (as shown), and the case where the SRF is locally connected to assisting SSP. In this latter case, the SRF communication (protocol flows B, C, G and H) would conform to the physical scenario shown in Figure 38.

The service 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. Consequently, no diagrams or tables are shown for this simple extension.

3.1.4 SDF application entity procedures

3.1.4.1 General

This subclause provides the definition of the SDF application entity (AE) procedures related to the SDF-SCF interface. The procedures are based on the use of Signalling System No. 7 (SS No.7).

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

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes 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 signalling message transport systems supporting the application layer structures defined.

3.1.4.2 Model and interfaces

The functional model of the AE-SDF is shown in Figure 41; the ASEs interface to TCAP to communicate with the SCF, and interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 41.

The interfaces shown in Figure 41 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-Primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of intelligent network application protocol (INAP) are defined in 2.

An instance of SDF FSM may be created if IN call handling is received from the SCF.

The SDF FSM handles the interaction with the SCF FSM.

3.1.4.3 Relationship between the SDF FSM and the maintenance function

The primitive interface between the SDF FSM and the maintenance functions is an internal interface and is not subject for standardization in CS-1.

3.1.4.4 SDF state transition model

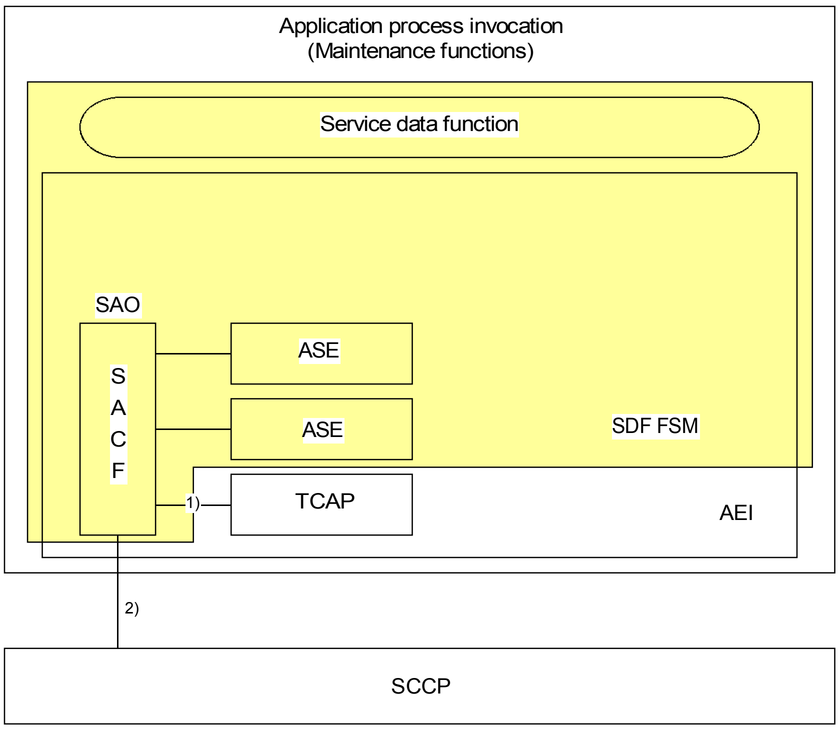
As far as the capabilities set 1 is concerned, the SDF's job is to (synchronously) respond to every request from the SCF. Hence, the respective finite state model (FSM), which is depicted in Figure 42, is trivial.

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 there is an error in a received operation, the maintenance functions are informed and the SDF FSM remains in the same state (i.e. idle state); depending on the class of the operation, the error could be reported by the SDF to the SCF using the appropriate component (see Recommendation Q.774).

In any state, if the dialogue with the SCF is terminated, then the SDF FSM returns to idle state after ensuring that any resources allocated to the call have been de-allocated.



AEI	Application entity invocation
SDF	Service data functions
FSM	Finite state machine
SACF	Single association control function
SAO	Single association object

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

NOTE – The SDF FSM includes several finite state machines.

FIGURE 41/Q.1218

Functional model of SDF AE

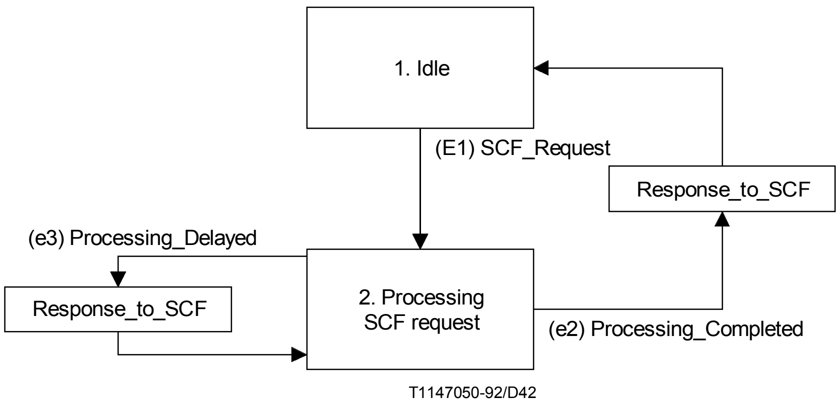


FIGURE 42/Q.1218

The SDF FSM

3.1.4.4.1 State 1: "Idle"

The only event accepted in this state is (E1) SCF_Request. This is an external event caused by the reception of Query or Update data operation from the SCF. This event causes a transition to state 2, Processing SCF request.

3.1.4.4.2 State 2: "Processing SCF request"

The events accepted in this state are as follows:

- (e2) Processing_Completed; and
- (e3) Processing_Delayed.

The event (e2) Processing_Completed is an internal event caused by the completion of the SCF request. This event causes the response to the Query or Update operation to be sent to the SCF and a transition to state 1, Idle.

The event (e3) Processing_Delayed is an internal event caused by the recognition of the delay to the SCF request. In order to inform the SCF about this situation, the SDF FSM sends the SDF response operation to the SCF and remains in the processing SCF request state.

It is important to keep in mind that for the network inter-working purposes, the SDF may return to the SCF a reference to another physical entity containing the requested information, instead of the requested information itself.

Therefore, obtaining that information may require more than one request from the SCF to different SDF-related physical entities.

Appendix I

Aspects of the intelligent network interface identified as "For Further Study" (FFS) relative to CS-1

(This appendix does not form an integral part of this Recommendation)

I.1 General

I.1.1 General consideration

This appendix includes call party handling and other issues which were considered to be incomplete when developing the Q.1218 intelligent network interface Recommendation CS-1. Although the material in this appendix is built upon CS-1, the procedures for these capabilities may be undefined and FFS relative to CS-1. The material is included in this appendix to provide some technical basis for future work.

I.1.2 Relationship to other appendices of the Q.1200-Series Recommendations

This appendix only applies to Q.1218 intelligent network interface Recommendation CS-1. Each of the Q.1200-Series Recommendations includes a specific appendix, if needed.

I.1.3 Format of Document

This Introduction provides an explanation of the purpose and scope of the appendix.

Subclause I.2 itemizes the operations.

Subclause I.3 itemizes the parameters.

Subclause I.4 is an ASN.1 module of the operations and parameters.

Subclause I.5 includes procedures for the operations

I.2 Operations

The operations listed in this subclause are in addition to the operations itemized in subclause 2.

I.2.1 Consideration applicable to all operations in this appendix

The following operations or aspects of the operations are FFS relative to CS-1. These operations rely on CS-1 capabilities for which the corresponding procedures are undefined. Therefore, they are included in this appendix for completeness.

The defined defaults may be incomplete (e.g. Leg ID) and may be dependent on the point in call (PIC).

I.2.2 Add party operation

I.2.2.1 Consideration

The difference between this operation and the attach operation needs to be clarified.

I.2.2.2 Description

SCF $\rightarrow$ SSF

This operation is used to perform the call processing actions to add all call party connections from one call to another call, then clear the first call (e.g. to create a conference call). From the perspective of the controlling party, this operation effectively bridges two calls.

I.2.3 Attach operation

I.2.3.1 Consideration

The difference between this operation and the add party operation needs to be clarified.

I.2.3.2 Description

SCF $\rightarrow$ SSF

This operation enables the SCF to request the SSF to include a leg in the current relationship instance. The leg is transferred from another relationship instance, from which it was removed using the detach operation. Notice that detach may also be executed after attach using the same absolute identifier.

I.2.4 Change parties operation

I.2.4.1 Description

SCF $\rightarrow$ SSF

This operation is used to perform the call processing actions to change a particular party connection from one call to another call. From the perspective of the particular call party, this operation effectively places the first call on hold and retrieves the associated call from hold.

I.2.5 Detach operation

I.2.5.1 Consideration

The difference between this operation and the release call party operation needs to be clarified.

I.2.5.2 Description

SCF $\rightarrow$ SSF

This operation enables the SCF to request the SSF to remove a leg from one relationship instance and to assign it an absolute (i.e. single network-wide) identifier (Correlation identifier), so that it can be transferred to another relationship instance, to which the leg was/will be attached by means of the attach operation using the same absolute identifier.

I.2.6 Hold call party connection operation

I.2.6.1 Description

SCF → SSF

This operation is used during the active phase of a call between two or more parties to put one party connection on hold.

I.2.7 Initiate call attempt operation – (*For case of more than 1 party*)

I.2.7.1 Consideration

This information flow is included in the main body of Recommendation Q.1214, for the case of creating a call to one call party. The IF is listed in this appendix for the case of creating a call to more than one party, in the same call, which for CS-1 is FFS.

I.2.7.2 Description

SCF → SSF

This operation is used to request the SSF to create a new call to one or more parties using address information provided by the SCF (e.g. predefined conference call, previous prompt and collect information). Any errors associated with performing this operation are returned.

I.2.8 Reconnect operation

I.2.8.1 Description

SCF → SSF

This operation is used to resume a held party to a call (inverse of hold call party connection).

I.2.9 Release call party connection operation

I.2.9.1 Consideration

The difference between this operation and the detach operation needs to be clarified.

I.2.9.2 Description

SCF → SSF

This operation is used to release a call party connection during a call between two or more parties.

I.3 Parameters

The parameters listed in this subclause are additional parameters for the operations itemized in subclause 2.

I.3.1 Considerations applicable to all parameters in this appendix

The following parameters are FFS realtive to CS-1. These parameters rely on CS-1 capabilities for which the corresponding procedures are undefined. Therefore, they are included in this appendix for completeness.

I.3.2 Leg Id created parameter (from Analyse information operation)

I.3.2.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

I.3.3 Leg Id created parameter (from connect operation)

I.3.3.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

I.3.4 Leg Id created parameter (from initiate call attempt operation)

I.3.4.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

I.3.5 Leg Id created parameter (from select facility operation)

I.3.5.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

I.3.6 Leg Id created parameter (from select route operation)

I.3.6.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

I.3.7 Leg 1 parameter (from initial DP operation)

I.3.7.1 Description

DEFAULT {aPARTY, pending}

Indicates call party information, as defined by a Leg object. This includes a LegID to reference each call party, and a LegStatus to indicate whether the call party is connected or not.

I.3.8 Leg 2 Parameter (from initial DP operation)

I.3.8.1 Description

OPTIONAL

Indicates call party information, as defined by a Leg Object. This includes a LegID to reference each call party, and a LegStatus to indicate whether the call party is connected or not.

I.3.9 Call Id Parameter

I.3.9.1 Description

Indicates an identifier to reference an instance of a Call accessible to the SCF. Refer to 4.2.2.1/Q.1214 for a description of call segment.

I.4 ASN.1 module of operations and parameters

The following modules describe the additional operations discussed in this appendix. The modules do not describe the modifications to the operations in subclause 2 to include the leg parameters discussed in subclause I.3.

I.4.1 Abstract syntax of the IN CS-1 application protocol – appendix

This subclause specifies additional abstract syntax for the IN CS-1 application protocol using abstract syntax notation one (ASN.1), defined in Recommendation X.208.

The material in this appendix is based on the appendix material in Reocmmendation Q.1214.

Operation types

```
IN-CS-1-Operations-appendix { ccitt recommendation q 1218 modules(0) cs-1-operations-app(4) version1(0) }

-- This module contains additional type definitions for IN CS-1 operations.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
            OPERATION,
            ERROR

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

-- TYPE DEFINITION FOR ADDITIONAL    IN CS-1    OPERATIONS

-- SCF-SSF operations

AddParty ::= OPERATION
        ARGUMENT
            AddPartyArg
        RESULT
            CallPartyHandlingResultsArg
        ERRORS {
            DataAlreadyExists,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SCF → SSF

-- This operation is used to perform the call processing actions to add all call party connections
-- from one Call to another Call, then clear the first Call (e.g. to create a conference call).
-- From the perspective of the controlling party, this operation effectively bridges two Calls.

Attach ::= OPERATION
        ARGUMENT
            AttachArg
        ERRORS {
            DataAlreadyExists,
            LegIDAlreadyAssigned,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            TooLate,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- SCF → SSF

-- This operation is used to attach two Calls.
```

```
ChangeParties ::= OPERATION
        ARGUMENT
           ChangePartiesArg
        RESULT
           CallPartyHandlingResultsArg
        ERRORS {
           DataAlreadyExists,
           MissingParameter,
           SystemFailure,
           TaskRefused,
           UnexpectedComponentSequence,
           UnexpectedDataValue,
           UnexpectedParameter
           }

-- SCF → SSF

-- This operation is used to perform the call processing actions to change a particular party
-- connection from one Call to an another Call. From the perspective of the particular call party,
-- this operation effectively places the first Call on hold and retrieves the associated Call from hold.

Detach ::= OPERATION
        ARGUMENT
           DetachArg
        ERRORS {
           DataAlreadyExists,
           MissingParameter,
           SystemFailure,
           TaskRefused,
           TooLate,
           UnexpectedComponentSequence,
           UnexpectedDataValue,
           UnexpectedParameter
           UnknownLegID
           }

-- SCF → SSF

-- This operation is used to detach two Calls.

HoldCallPartyConnection ::= OPERATION
        ARGUMENT
           HoldCallPartyConnectionArg
        RESULT
           CallPartyHandlingResultsArg
        ERRORS {
           DataUnavailable,
           MissingParameter,
           SystemFailure,
           TaskRefused,
           UnexpectedComponentSequence,
           UnexpectedDataValue,
           UnexpectedParameter
           }

-- SCF → SSF

-- This operation is used during the active phase of a call between two or more parties to put one
-- party connection on hold.

Reconnect ::= OPERATION
        ARGUMENT
           ReconnectArg
        RESULT
           CallPartyHandlingResultsArg
        ERRORS {
           DataAlreadyExists,
           MissingParameter,
           SystemFailure,
           TaskRefused,
           UnexpectedComponentSequence,
           UnexpectedDataValue,
           UnexpectedParameter
           }
```

-- SCF → SSF

-- This operation is used to resume a held party to a call (inverse of HoldCallPartyConnection).

```
ReleaseCallPartyConnection ::= OPERATION
        ARGUMENT
            ReleaseCallPartyConnectionArg
        RESULT
            CallPartyHandlingResultsArg
        ERRORS {
            DataAlreadyExists,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- SCF → SSF

-- This operation is used to release a call party connection during a call between two or more parties.

END

IN-CS-1-Errors-appendix { ccitt recommendation q 1218 modules(0) cs-1-errors-app(5) version1(0) }

-- This module contains additional type definitions for the IN CS-1 errors.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

```
IMPORTS
            ERROR
```

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

-- TYPE DEFINITION FOR ***IN CS-1*** *ERRORS*

LegIDAlreadyAssigned ::= ERROR

-- Indicates that a legID has already been assigned with the requested value.

TooLate ::= ERROR

-- Indicates that the operation could not be performed in a timely manner.

UnknownLegID ::= ERROR

-- Indicates that the legID does not exist.

END

Data types

IN-CS-1-DataTypes-appendix { ccitt recommendation q 1218 modules(0) cs-1-datatypes-app(6) version1(0) }

-- This module contains additional type definitions for the IN CS-1 data types.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- TYPE DEFINITION FOR ADDITIONAL ***IN CS-1*** *DATA TYPES*

-- Argument Data Types

```
AddPartyArg ::= SEQUENCE {
        originalCallID              [0]  CallID                   OPTIONAL,
        destinationCallID           [1]  CallID                   OPTIONAL,
        }
```

-- OPTIONAL denotes network operator specific use.

```
AttachArg ::= SEQUENCE {
        newLegID                    [0]  LegID                    OPTIONAL,
        correlationidentifier       [1]  CorrelationID            OPTIONAL,
        }
```

-- OPTIONAL denotes network operator specific use.

CallPartyHandlingResultsArg ::= SEQUENCE OF LegInformation

```
ChangePartiesArg ::= SEQUENCE {
        callID                      [0]  CallID                   OPTIONAL,
        targetCallID                [1]  CallID,
        legToBeConnectedID          [2]  LegID
        }
```

-- OPTIONAL denotes network operator specific use.

```
DetachArg ::= SEQUENCE {
        legToBeDetached             [0]  LegID                    OPTIONAL,
        correlationidentifier       [1]  CorrelationID            OPTIONAL,
        }
```

-- OPTIONAL denotes network operator specific use.

```
HoldCallPartyConnectionArg ::= SEQUENCE {
        callID                      [0]  CallID                   OPTIONAL,
        legID                       [1]  LegID
        }
```

-- OPTIONAL denotes network operator specific use.

```
ReconnectArg ::= SEQUENCE {
        callID                      [0]  CallID                   OPTIONAL,
        heldLegID                   [1]  LegID
        }
```

-- OPTIONAL denotes network operator specific use.

```
ReleaseCallPartyConnectionArg ::= SEQUENCE {
        legToBeReleased             [0]  LegID,
        callID                      [1]  CallID                   OPTIONAL,
        releaseCause                [2]  Cause                    OPTIONAL,
        }
```

-- OPTIONAL denotes network operator specific use.Common Data Types

CallID ::= INTEGER

-- Indicates an identifier to reference an instance of a Call accessible to the SCF. Refer to 4.2.2.1/Q.1214
-- for a description of Call Segment.

Cause ::= OCTET STRING

-- Indicates the cause for interface related information. Refer to the Q.763 Cause parameter
-- for encoding.

```
LegInformation ::= SEQUENCE {
        legID                       [0]  LegID,
        legStatus                   [1]  LegStatus
        }
```

-- Indicates call party information, as defined by a Leg object. This includes a LegID to reference each
-- call party, and a LegStatus to indicate whether the call party is connected or not.

```
LegStatus ::= ENUMERATED {
                connected(0),
                unconnected(1),
                pending(2),
                interacting(3)          -- user connected to a resource
        }
-- Indicates the state of the call party.

END

Application Protocol (Operation and Error Codes)

IN-CS1-Codes-appendix { ccitt recommendation q 1218 modules(0) cs-1-codes-app(7) version1(0) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- OPERATION AND ERROR CODE DEFINITION

-- code point values are for further study.the operations are grouped by the identified ASEs.
-- Call party handling ASE

addParty                            AddParty ::= ffs

changeParties                       ChangeParties ::= ffs

holdCallPartyConnection             HoldCallPartyConnection ::= ffs

reconnect                           Reconnect ::= ffs

releaseCallPartyConnection          ReleaseCallPartyConnection ::= ffs

-- Attach ASE

attach                              Attach ::= ffs

detach                              Detach ::= ffs

END
```

I.5 Procedures

PRINCIPLE is as follows:

The SSF procedures are illustrated by means of finite state machines (FSMs).

The (call level) finite state machine, presently described in this Recommendation, consists of several states, which represent different states of the call.

In some of the call states, another level of finite state machine(s) exists, which will have to be described, and which represents the state of a connection to one of the parties. This FSN is named “leg level FSM”.

Such a “leg level FSM” is “born out” or “dies out” of some of the call level FSM transitions. any “event” (e.g. an operation received) causes a state transition of all relevant FSMs.

The Leg states used are the possible values of the parameter “LegStatus”, are the following:

- “IDLE”: connection to the party does not exist.
- “PENDING”: party not connected, and under creation process.
- “UNCONNECTED”: party not connected, but in a stable state.
- “CONNECTED”: party connected to another party.
- “INTERACTING”: party connected to an SRF.

Printed in Switzerland

Geneva, 1993