ETSI

ETSI TECHNICAL REPORT

ETR 024

May 1991

UDC:

Key words: .

Signalling Protocols & Switching (SPS); Intelligent Networks switching aspects

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Blank page

Foreword

ETSI Technical Reports (ETRs) are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim - European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or I-ETS.

This ETR has been produced by the Signalling Protocols & Switching (SPS) Technical Committee of the European Telecommunications Standards Institute (ETSI).

Introduction

The information contained in this report reflects initial studies on switching aspects in Intelligent Networks (IN). The objectives of these studies were to develop a general understanding of IN switching aspects with a view to creating a basis for future IN studies in SPS3.

The studies were conducted prior to recent efforts in ETSl and CCITT which focus on the first standardisation phase targeted for 1992 and is known as the so-called Capability Set Number 1 (CS1).

Some of the information in the report may be relevant for both Capability Set Number 1 and/or longer term Capability Sets. Other information may be either beyond the scope of CS1, or may become obsolete as current studies on IN are progressed.

This report is divided in two parts:

> Part 1 deals with the Connection Control Model (CCM). The contents of sections 1 to 3 is related to the functional plane of the IN conceptual model developed by ETSI STC/NA6;
>
> Part 2 identifies different functional entities of IN especially the Service Switching and Call Control functional entities (SSF & CCF). A detailed study of the logical behaviour of SSF and CCF is started. The concept of trigger table are introduced and the behaviour of connection segment objects is described. Different chapters have reached different degrees of detail.

The result present in this technical report is an intermediate result and should be considered as a state-of-the-art at the end of 1990.

Blank page

Part 1: The Connection Control Model (CCM)

Blank page

OBJECTIVES & DESCRIPTION OF THE MODELLING

1.1 Objectives

The Connection Control Model is used to describe the interaction between functional entities (principally the SSF and SCF) in the functional plane of the IN conceptual model.

The Connection Control Model establishes a framework for the interaction between the SSF and SCF in a service- and vendor implementation-independent environment.

The CCM should only be used in order to provide a description of the SSF/CCF functions (triggers, events, operations, etc.) as seen from outside the SSF/CCF. It is not intended to place any constraint on implementations.

1.2 The Connection Control Model (CCM)

The following is a general description of the Connection Control Model (CCM). The full description is contained in the Draft Technical Report ETSI DTR/NA-6001 produced by NA6.

The CCM is a model of the interactions between the Service Control Function and the Service Switching function in an Intelligent Network (IN), in order to provide telecommunications and supplementary services related to calls in an IN-structured network.

Therefore, the CCM is centred around the concept of a "call". A call can be defined as a temporary relation between two or more users of a telecommunications network, for the provision of telecommunications services and possible associated supplementary services (a user can be a human, a machine, a system, etc.). A call, in order to be established, maintained and released, involves resources of the telecommunications network.

An "IN call" can be defined as a call that involves resources belonging to an Intelligent Network architecture.

Typically, the life-cycle of an IN call is composed of the following phases:

a) a call request is issued by a telecommunications network user, and non-IN switch-based processes are activated in order to fulfill such a request

b) a Trigger Point is encountered and the Trigger Condition is fulfilled: subsequent call processing implies the interaction between IN entities (SSF, SCF, etc.)<1>.

<1> Phases a) and b) are missing if the service is triggered by the Service Control Function (e.g. "wake-up service"). In this case the SCF creates the socket autonomously and phase c) follows

c) a socket (or more than one) is opened and interactions between SSF and SCF occur through this socket

d) call processing can proceed without the interaction between IN entities, therefore the socket is closed and non-IN switch-based call processing is resumed

e) the call is finally terminated.

The Socket Models apply to phase c) of the call. They can be viewed as a "window" between the two functional blocks (SSF and SCF), through which they can see each other and interact in terms of pre-defined objects and operations on the objects.

Some interactions between SSF and SCF will not be associated to any specific call. To distinguish between call-related and non call-related interactions, different "socket types" are used, i.e. connection control socket, data management socket, status monitoring socket, traffic management socket and resource control socket.

The Connection Segment Relationship Model also applies to phase c) of the call, to cater for the interactions between different sockets and different service features (both provided in an IN fashion and provided locally by switch-based processes) related to the same call.

The Basic Call Model applies to phases b) and d) of the call. It is a model of the logical states of a call and of the points (states or transitions between states) in which the socket "window" can be opened and closed.

The three models together form the Connection Control Model.

1.2.1 The Basic Call Model (BCM)

Utility of the BCM: its main purpose is to limit, to an acceptable degree, the flexibility of choice of Trigger Points (TP's) within a call life-cycle. Once the set of allowed points is specified, additional information can be associated to each of them, to form the Trigger Table.

Consistency of the BCM with the other models and with call processing in general: at the time of socket opening, sufficient information must be sent from SSF to SCF in order to allow for a correct sequence of call processing. During the interactions through the socket(s), consistency depends on the correctness of modelling objects/operations and of service logic. At the time of socket closing a correct resumption of basic switch processing must be ensured.

The BCM is described in detail in Part II of this Technical Report.

2 TECHNICAL DETAILS OF THE CCM

The general description of the use of the different socket types in the Connection Control Model is described in the NA6 Draft Technical Report ETSI DTR/NA-6001.

This section contains the requirements on the parameters and primitives related to the objects available in the different socket types, in order to constitute an input for the consequent definition of application protocol elements.

2.1 Connection Control Socket

The connection control (CC) socket is used by the SCF to operate connection control related objects in a CCF/SSF, namely legs and connection points.

The primitives on the objects and the corresponding parameters are described in detail in these sections.

Note(s): Error replies are not specified.

2.1.1 CC-Provide-Instruction

The SSF uses this primitive to indicate service activation to the SCF, to request instructions from it.

The CC-PROVIDE-INSTRUCTION includes parameter(s) to describe the present content of the socket; i.e. the objects which exist within the socket namely leg and connection point identifiers, as well as information identifying the trigger condition (service activated, event which triggers the service).

Other parameters like CalledPartyNumber, CallingPartyNumber, CallingPartyCategory, UserServiceInformation, CUGInterlockCode, RedirectionInformation, RedirectingNumber, OriginalCalledNumber, which are all similar to ISUP parameters, indicate detailed characteristics of the leg(s) which exist(s) in the socket.

Information that needs to be exported from SSF to SCF is defined related to trigger conditions.

2.1.2 CC-CREATE a leg

The SCF uses this primitive to request the creation of a leg by the SSF to an addressable entity.

The CC-CREATE includes the following parameters:

- “leg identifier” to identify the leg to create

- “addressable entity identifier” to identify the addressable entity.

- "leg attributes”, like CallingPartyNumber, CallingPartyCategory, UserServiceInformation, CUGInterlockCode, RedirectionInformation, RedirectingNumber, OriginalCalledNumber, similar to ISUP parameters, to indicate detailed characteristics of the leg to create.

- "other leg identifier”, identifying a leg already existing whithin the socket, indicating that this legs may be joined with the created one in the near future.

- "completion condition”, to indicate, when the CC-CREATE can be considered as completed by the SSF (when the next operation on this leg can start).

The “other leg identifier” parameter, identifies a leg already existing whithin the socket. When this parameter is present, the SSF derives missing information for the created leg (e.g. CallingPartyNumber, . ..) from this other leg. Otherwise, default values (t.b.d.) are used.

Possible values of “completion condition” may be: “Local processing complete”, “alerting”, “answered”. It indicates when the next operation on the created leg can start.

Note(s): Issues such as the need to distinguish between different types of adressable entities towards which legs are created (Could that be a leg characteristic? Should there be any difference between a leg to a “normal” party and another to an SRF?) are not specified.

2.1.3 CC-FREE a leg

The SCF uses this primitive to request the SSF to release a leg and all associated resources.

The CC-FREE includes the following parameters:

- “leg identifier” to identify the leg which must be freed

- “release cause” (similar to ISUP CauseIndicators)

- “when” to indicate whether the leg must be released immediately, any other operation on this leg being stopped, or only after the completion of ongoing operation(s) on this leg

The “leg identifier” parameter, which identifies the leg to release, it must identify an unjoined leg.

The parameter used to identify the release cause is “translated” by the SSF, depending on the signalling system used for the leg.

Another parameter indicates whether the leg must be released immediately, any other operation on this leg being stopped, or only after the completion of ongoing operation(s) on this leg.

2.1.4 CC-JOIN legs

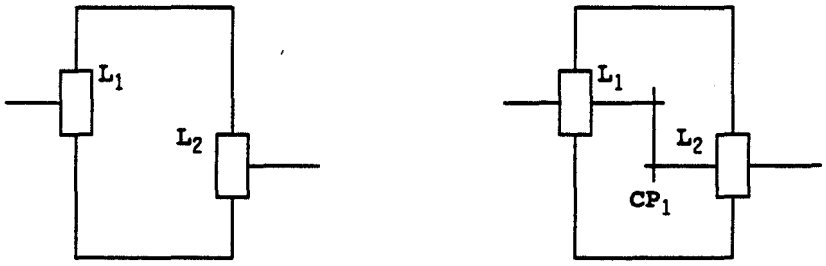
The SCF uses this primitive to request the SSF to link either one or two leg(e) or a connection point to a connection point and to through-connect the corresponding physical resources.

The CC-JOIN includes the following parameters:

- “leg identifier”(s) to identify the leg(s) concerned;

- “connection point identifier”(s) to identify the CP(s) concerned

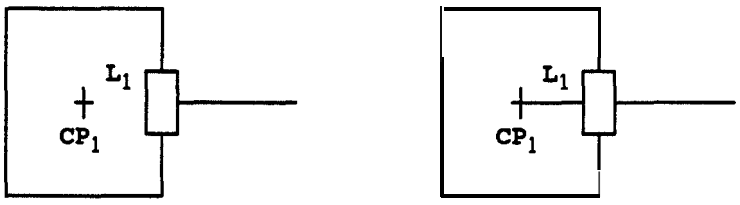
When used to connect two separate legs on a non existing connection point, a parameter “leg identifier” identifies the first leg, a parameter “other leg identifier” identifies the other leg to connect and a parameter “CP identifier” identifies the connection point.



before after

Figure 1. socket before and after CC-JOIN (first case)

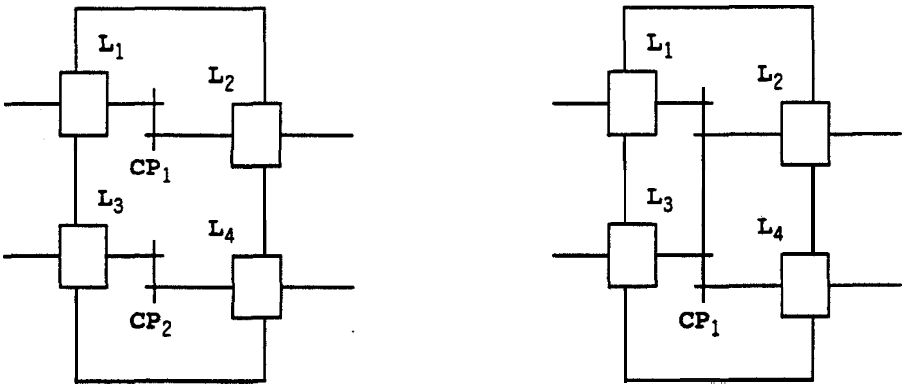
When used to connect a leg to an existing connection point, a parameter “leg identifier” identifies the leg and a parameter “CP identifier” identifies the connection point. In this case, several legs may already be connected to the connection point.



before after

Figure 2. socket before and after CC_JOIN (second case)

When use to connect two existing connection points together, two parameters “CP identifier” identify the connection points. As a result of this operation, all legs connected to both connection points end up connected to a single connection point.



before after

Figure 3. socket before and after CC-JOIN

2.1.5 CC-SPLIT a leg

The SCF uses this primitive to request the SSF to separate a leg from a connection point, or all the legs from a connection point, discarding the connection point.

The CC-SPLIT includes the following parameters:

- “leg identifier" to identify the leg concerned, or

- "connection point identifier” to identify the CP concerned.

When only a connection point identification is present, all legs connected to it are split off, and the connection point is released.

When only a leg identification is present, only the indicated leg is split off from the connection point on which it was connected. (a leg can only be connected to one connection point at a given time).

Note: Application of this primitive for the splitting of one CP into two separate CP’s is not specified here.

2.1.6 CC–MONITOR events on a leg

The SCF uses this primitive to request the SSF to assign or update the “event processing mode” to some or all events on one leg in the SSF. Event mode may be: transparent, intercepted, duplicated, ignored.

The CC-MONITOR includes the following parameters:

- “leg identifier”, to identify the leg which is concerned

- for each event: “event mode”, to identify in which processing mode the SSF must handle the event.

- “other leg identifier”, to identify the leg on which transparent or duplicated event must be reflected.

The “leg identifier” parameter identifies the leg concerned. Another parameter defines in which processing mode the SSF must handle events.

If an event is asked to be in transparent or duplicated mode, another “leg identifier” parameter must indicate on which other leg the event must be reflected (sent/transmitted).

2.1.7 CC-GENERATE an event on a leg

The SCF uses this primitive to request the SSF to generate on a leg the signalling message corresponding to an event. Information relative to this message is given by the SCF in a way. It is up to the SSF to translate it in the right signalling system.

The CC-GENERATE includes the following parameters:

- “leg identifier” to identify the leg on which the event has to be generated

- “event identifier”, to indicate which event must be generated

- “event parameter”, to describe the parameter of this event.

The “leg identifier” parameter to identifies the leg on which the event must be generated. The “event descriptor” parameter describes the event.

Limitations on which events can be generated are not specified. However, events that can be “generated” by other operations are excluded (eg: CC-GENERATE Release is excluded, as CC-FREE is available).

2.1.8 CC-EVENT received on a leg

The SSF uses this primitive to report information on events to the SCF. This event has been previously assigned one of the two modes: intercepted or duplicated.

The CC-EVENT includes the following parameters:

- “leg identifier” to identify the leg on which the event occurs

- “event identifier” to indicate which event was received

- “event parameter” to describe the parameters of this event

- “event date" to indicate when the event occurs.

2.1.9 CC-SEND-RECEIVE information on a leg

The SCF uses this primitive to request the SSF to send and/or receive specific information to and/or from a call participant.

The CC-SEND-RECEIVE includes the following parameters:

- “leg identifier” to identify the leg on which the information has to be exchanged

- InformationToSend

  This subparameter contains the specification of the information to be send Tone Identity Announcement Identity other Identities are not specified

- SendStopOptions

  This subparamater contains one of the following stop options: NumberOfRepetitions Duration InformationReceived NextOperationReceived

- InformationToReceive

  This subparameter contains the number of digits to be received

- ReceiveStopOpt.

  This subparameter contains the receive parameters: InitialCharacterTimeOut InternalCharacterTimeOut TotalElapsedTimeOut EndOfInputDelimiter AllDigitsReceived Others are not specified.

2.1.10 CC-RESUME basic call

The SCF uses this primitive to resume object control to the SSF. With this operation, the SCF gives control back to the SSF on some objects, like leg(s) or connection point(s), which will then be handled with the basic call control.

The CC-RESUME includes the following parameters:

- “connection point identifier”

The SCF can only give back control on a connection point, to which two legs are connected.

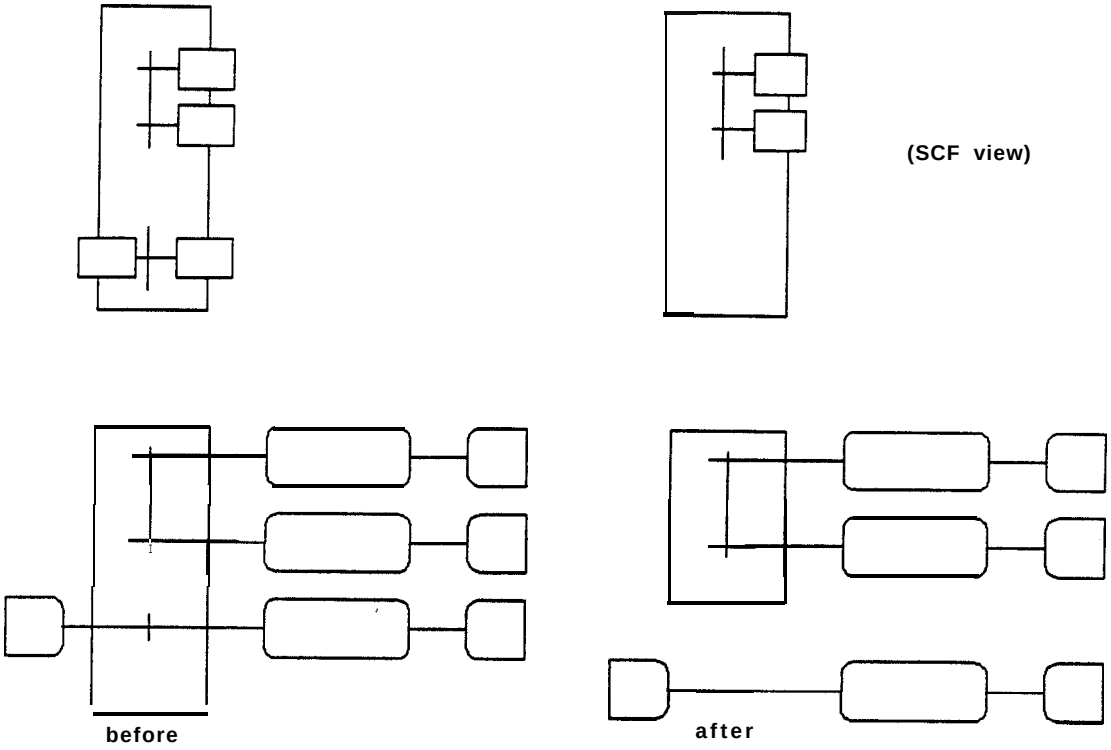


Figure 4. example of CC_RESUME

2.1.11 CC-DETACH a leg

The SCF uses this primitive to request the SSF to remove a leg from the socket and to asign it an absolute (i.e. unique network wide) reference, so that it can be transferred to another socket instance, to which the seg was/will be attached by means of the CC-ATTACH primitive using the same absolute reference.

The CC-DETACH includes the following parameters:

- "leg identifier", to identify the leg to detach

- "absolute reference", which allows to find the socket where the leg has to be attached.

Only unjoined legs can be detached.

If a CC-ATTACH with the same absolute reference was already received by the SSF, the concerned leg is transferred from one socket to the other. Otherwise, the SSF sets a timer. When the timer elapses, if the leg transfer has not been done, an error response is given to the SCF for this CC-DETACH operation.

2.1.12 CC-ATTACH a leg

The SCF uses this primitive to request the SSF to include a leg in the current socket. The leg is transferred from another socket, to which the service logic will send (or has already sent) a CC-DETACH primitive with the same absolute reference.

The CC-ATTACH includes the following parameters:

- “leg identifier”, will identify the leg after it has been attached

- “absolute reference”, which allows to find the leg to attach.

If a CC-DETACH with the same absolute reference was already received by the SSF, the concerned leg is transferred from one socket to the other. Otherwise, the SSF sets a timer. When the timer elapses, if the leg transfer has not been done, an error response is given to the SCF for this CC-ATTACH primitive.

After CC-ATTACH, all events on the leg are in the intercept mode until the next CC-MONITOR.

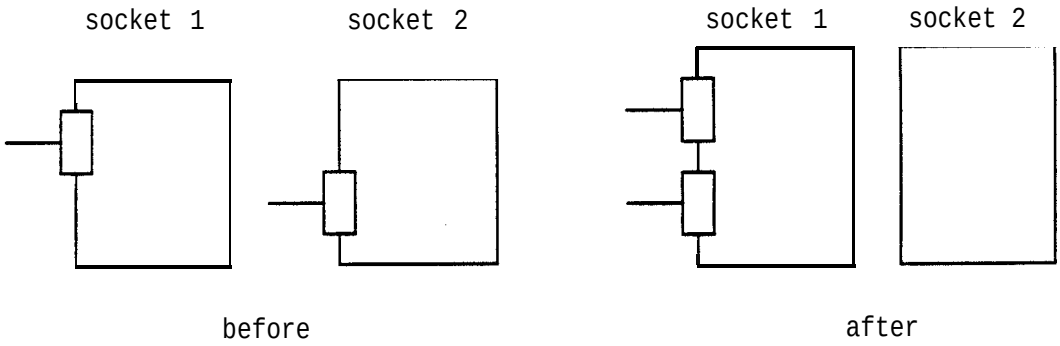


Figure 5. example of CC_ATTACH in socket 1 and CC_DETACH in socket 2

2.1.13 CC-MODIFY a leg

The SCF uses this primitive to request the SSF to modify some characteristics of an existing leg in the current socket (eg. to change the charging rate applied to the leg, or to change the bandwith associated with the leg, . ..).

2.1.14 CC-SEND information on a connection point

The SCF uses this primitive to request the SSF to broadcast information (eg. tones or announcements) to all legs connected to a connection point.

The CC-SEND includes the following parameters:

- "connection point identifier", to identify the connection point to which the broadcast applies

- "information to send", to identify the information which has to be broadcast.

2.2 Data Management socket

This section contains a detailed description of primitives in the DM-socket and related parameters.

2.2.1 DM-Create primitive

The SCF uses this primitive to create a new data item in the SSF. This does not include the creation of a new table but only the creation of a new record in en existing table. The primitive should include the following parameters

DataItemId
DataItemType
CounterIncrementUnit
DataItemInformation
BufferSize
BufferBehaviour

2.2.2 DM-Delete Primitive

The SCF uses this primitive to delete a data item in the SSF. The primitive should include the following parameters:

DataItemId

2.2.3 DM-Modify primitive

The SCF uses this primitive to update an existing data item. The primitive should include the following parameters:

DataItemId
DataItemInformation
CounterIncrementValue
CounterIncrementUnit

2.2.4 DM-Poll primitive

The SCF uses this primitive to interrogate the value of a given data item. The primitive should include the following parameters:

DataItemId
CounterResetIndicator
DataItemInformation

2.2.5 DM-Event primitive

The SSF uses this primitive to inform the SCF of either a reception of a DM-Modify primitive from another IN entity, or an updating of a data item due to internal reasons or to an end user - the primitive should include the following parameters:

DataItemId
DataItemInformation
EventCause

2.2.6 DM-Monitor primitive

The SCF uses this primitive to specify the treatment of a data modification attempt. This primitive should include the following parameters:

DataItemId
MonitorMode

2.2.7 Parameters used in primitives of the Data Management Socket

DataItemId : This parameter specifies the data item identifier as it is known by the SSF or the SCF. The parameter should specify the detailed addressing of the data to be accessed. (e.g. up to the specification of the field within a table entry.

DataItemType : This parameter specifies the type of the DataItemIdentifier

Possibilities:
- Record
- Counter
- Buffer

BufferSize : This parameter specifies the storage capacity in case the DataItemIdentifier specifies a buffer.

BufferBehaviour : This parameter specifies the action to be taken in case the buffer is full:

Possibilities:
- override the oldest value
- override the latest value
- reject near storage requests

Counter IncrementUnit : This parameter specifies the value by where a

counter should be incremented each time a counter increment request is received.

Counter IncrementValue : This parameter specifies the value by which a specified counter should be incremented. This parameter specifies a single action.

CounterReset Indicator : This parameter specifies that the counter after being read should be reset.

EventCause : This parameter specifies the reason for an event reporting.

Possibilities: EndUser (line identity)
Internal
DM-Modify primitive (IN entity address)

MonitorMode : This parameter specifies the action to be taken on the specified DataItemIdentity

Possible actions to be taken when an attempt to access the data is made:

- lock without reporting (similar to Discard in the CCSocket)
- lock with reporting (similar to Intercept in the CCSocket)
- unlock without reporting (similar to Transparent in the CCSocket)
- unlock with reporting (similar to Duplicate in the CCSocket)

DataItemInformation : This parameter specifies the contents of the DataItemIdentity

2.3 Status Monitoring Socket

In the following primitives on the objects contained in the Status Monitoring Socket are described in detail.

2.3.1 SM- Start Status Monitoring Primitive

The SCF application uses this primitive to request the monitoring and reporting of busy/free status changes of a specified object (resource). Note that the possibility to request the reporting of the next status change and the reservation of the object at the moment the state changes to free is not specified.

The primitive should include the following parameters:

MonitoringReference: This parameter contains a reference allocated by the SCF. This reference is used by the SCF to correlate the replies from the SSF/SRF to the requested monitoring. This parameter is to be used in all further communication (by the SCF as well as by the SSF/SRF) related to the requested monitoring.

ResourceType: This parameter identifies the type of resource for which the monitoring request should be executed.

ResourceIdentity: This parameter contains the resource identification as known by the SSF/SRF.

Possibilities:
- E.164 number
- DTMF Receiver
- Announcement Sender
- Other are FFS

MonitorIndicator: This paramater contains the request type of monitoring

Continuously
Next Change

2.3.2 SM - Cancel Status Monitoring Primitive

The SCF application uses this primitive to request the cancellation of a previously requested monitoring.

The primitive should include the following parameters:

MonitoringReference

2.3.3 SM - Poll Status Primitive

The SCF application uses this primitive to request the present state of the specified object.

The primitive should include the following parameters:

ResourceType

ResourceIdentity

The reply on this primitive contains the present state (i.e Busy or Free).

2.3.4 SM- Event Status Primitive

This primitive is used by the SSF/SRF to report a state change on an object, previously placed under supervision by the SCF. The state changes to be reported are: from Busy to Free or from Free to Busy.

The primitive should include the following parameters:

MonitoringReference: The same parameter value is used as received in the request.

ResourceState: This parameter contains the resource state of the resource placed under monitoring by the SCF.

2.4 Traffic Management Socket

In the following, primitives and related parameters on the objects contained in the Traffic Management Socket are described in detail.

2.4.1 TM—CREATE primitive

The TM-CREATE primitive is used by the SCF to request the creation of 1 logical filter object in the socket. This implies, in the SSF, the activation of the corresponding physical filter. The primitive may include<1> the following parameters: filter scope, filter type, filter severity, filter duration and call treatment.

The combination of filter scope and filter type may be used as an identifier of the filter.

scope This parameter is composed of three sub-parameters, i.e. Trigger Check Point (TCP), SCF Addressing Information (SAI) and SCF Export Information (SEI). It must be noted that such sub-parameters are in fact characteristics of IN calls, as they define the specific subset of IN calls passing through the SSF onto which the filtering mechanism (defined by the last 3 parameters) must be applied. Moreover, these sub-parameters are information contained in each entry of the Trigger Table. The function of each parameter is given below. Their formats and the sets of allowed values are defined in the Trigger Table.

TCP This sub-parameter specifies a subset of IN calls to which the filter applies, i.e. all those calls that hit the trigger at the specified Trigger Check Point. It is proposed that for the purpose of IN TM only the TCP'S in the originating portion of the call be considered, e.g. the TCP after digit analysis, but not the TCP for busy destination. This is however not specified.

<1> optionality vs. mandatoriness for each parameter or sub-parameter is not specified

SAI This sub-parameter specifies a subset of IN calls to which the filter applies, i.e. all those calls that generate an interaction with the specified SCF. An example of SAI value may be: the SS7 Global Title corresponding to a given SCP.

SEI This sub-parameter specifies a subset of IN calls to which the filter applies, i.e. all those calls that imply the transfer of the specified information from the SSF to the SCF, at the very beginning of the interaction between them for call handling. An example of SEI value is the calling line identity.

Type This parameter specifies the type of decision that leads to filter activation. It is a binary parameter with the values: Manual/Automatic.

Severity This parameter specifies the number of calls that are blocked by the filter, vs. calls that are let through. This is a logical indication and places no constraints on the actual filtering mechanism adopted by the SSF to filter the calls (e.g. call gapping). Allowable values for this parameter are not specified.

Duration This parameter specifies the span of time elapsing between activation of the filter by the SSF, on request of the SCF, and de-activation of the filter autonomously by the SSF, supposing no other operation is issued in the meantime by the SCF on the same filter. Allowable values for this parameter are not specified.

Treatment This parameter specifies the treatment that the SSF has to apply to all the IN calls that are blocked by the filter. In general such calls can only receive either a tone or a verbal announcement (other treatments are not specified), thus the parameter is a logical identifier of tones and announcements. This parameter may also specify whether filtered calls should be counted or not. Allowable values for this parameter are not specified.

Other parameters are for further study (e.g. the identity of the SCF that activates the filter).

2.4.2 TM-CANCEL primitive

The TM-CANCEL primitive is used by the SCF to request the immediate deletion of one or more logical filter object(s) in the socket. This implies, in the SSF, the de-activation of the corresponding physical filters. The primitive may include 2 parameters, i.e. filter scope and filter type (these are the same parameters defined for the TM-START primitive).

When both parameters are specified, 0 or 1 filter object is deleted, depending on whether an object exists with those parameter values. The absence of either of the parameters implies the deletion of all (0 to N) filter objects in the socket that match with the specified parameter value. The absence of both parameters implies the deletion of all (0 to N) filter objects in the socket, i.e. the de-activation of all the filters in the SSF that had previously been activated by that SCF.

2.4.3 TM-EVENT primitive

The TM-EVENT primitive is used by the SSF to signal to the SCF the occurrence of a specific event on a filter.

Example of events can be the following:

- the deletion of a logical filter object indicating filter de-activation by the SSF because of the expiration of the related duration timer

- the creation modification or deletion of filter objects in different sockets (even with different SCF's) that correspond to the same physical filter in the SSF.

The list of possible events, as well as the detailed specification of the EVENT operation parameters, is not specified. Some events may be accounted for in responses to the other operations.

2.4.4 TM-MODIFY primitive

The TM-MODIFY primitive is used by the SCF to request the modification of one or more of the parameter values in already existing filter object(s). This implies, in the SSF, the modification of the corresponding active physical filter(s). The operation may include the same parameters as defined for the START primitive.

The possibility of merging this operation with the CREATE primitive is not specified.

2.5 Resource Control Socket

This socket type is only used between SCF and SRF. The reason why this socket type is described is because there is a view that the SRF is a specific instance of the SSF.

In the following, primitives on the objects contained in the Resource Control Socket are described in detail.

2.5.1 RC-RESERVE RESOURCE Primitive

The SCF uses this primitive to request the reservation of a specialised resource (e.g. conference bridge, speech recognition devices...), allocated at a SRF.

The resources used for interactive communication (e.g. sending of announcements, reception of digits) between addressable entities and the network are normally not "reserved" by the SCF, but requested via the Send&Receive operation (see further) during the execution of service logic.

The primitive should include the following parameters:

ResourceType This parameter specifies the type of the resource, as it is known at the SRF.

- For each resource type further information is included to specify the resource characteristics e.g. max number of legs that can be connected to a conference bridge.

Time This parameter specifies the time when the resource should become available.

Duration This parameter specifies for how long the resource should remain available.

The reply on this primitive is a ResourceAddress, to be used by the SCF when addressing (via primitives) the reserved resource.

Note:

- When resource management is performed in a central point in the network, reservation of resources must not be executed any more at SRF level (e.g. distributed).

  When there is no central resource management, however, the reserve resource primitive is necessary because SCF's, located at different physical locations may request independently of each other the reservation of resources to the same SRF.

2.5.2 RC—ALLOCATE RESOURCE Primitive

The SCF uses this primitive to request the allocation of a resource. This allocation is necessary to provide the SCF with the necessary routing information to set up a transport connection between the SSF and the SRF.

This primitive should include the following parameter:

ResourceType The same parameter structure as described in the reserve resource primitive is used. The values of the resource type are due to this primitive extended with:

- DTMF receiver
- Announcement Sender
- Other resources are not specified

The reply on this primitive is a ResourceAddress known by the public network.

This address is used by the SCF:

- to request (via a SSF) the set up a transport path to the allocated resource in the SRF,
- to request actions (via primitives) for the allocated resource in the corresponding SRF.

2.5.3 RC-RELEASE RESOURCE Primitive

The SCF uses this primitive to request the release of the allocated/ reserved resource.

This primitive should include the following parameters:

ResourceAddress This is the same parameter as replied by the SRF as result of the execution of the Reserve Resource Primitive or the Allocate Resource Primitive.

2.5.4 RC-SEND&RECEIVE Primitive

The SCF uses this primitive to request the sending/reception of information via the specified resource.

This information may either be inband (e.g. tone, announcement) or out of band (e.g. D-Channel message).

This Primitive should include the following parameters:

ResourceAddress

SendPortion This parameter contains the subparameters InformationToSend and the SendStopOptions

ReceivePortion — This parameter contains the subparameters InformationToReceive and the ReceiveStopOptions

InformationToSend — This subparameter contains the specification of the information to be send

- Tone Identity
- Announcement Identity
- other Identities are not specified

SendStopOptions — This subparameter contains one of the following stop options:

- NumberOfRepetitions
- Duration
- InformationReceived
- NextOperationReceived

InformationToRec — This subparameter contains the number of digits to be received

ReceiveStopOpt — This subparameter contains the receive parameters:

- InitialCharacterTimeOut
- InternalCharacterTimeOut
- TotalElapsedTimeOut
- EndOfInputDelimiter
- AllDigitsReceived
- Other parameters are not specified

3 TECHNICAL BACKGROUND AND RATIONALE

This section contains some rationale and background information in order to clarify the technical choices and the models described in the previous sections.

3.1 Connection Control Socket

This is the connection control-oriented socket through with the SCF and the SSF may communicate each other for the establishment and control of IN service calls on a per a call basis. The SSF may only invoke SCF-based IN features or services through an instance of this socket type (the SSF opens the socket). However some of the SCF-based services or features can be triggered from inside the SCF (e.g., a "wake-up" service) with no request from SSF (the SCF opens the socket).

Neither the SCF nor the SSF will use this socket type to invoke non-connection control oriented features or network functions.

Logical objects in a connection control socket are legs and connection points. These objects only can be manipulated through a connection control socket. From the perspective of an instance of any SCF-based feature the scope and lifetime of the legs and connection points created for an instance of this socket type is limited to the scope and lifetime of the socket instance itself.

3.1.1 Objects in the Connection control socket

Legs are created explicitly by primitives from SCF or by the SSF, autonomously, at service creation time. A connection point is implicitly created when two existing legs are joined. In order to join legs connection points are necessary.

A leg is an abstraction of a communication path between the network entity supporting the SSF functionality and some effective network address (e.g. an end-user or an information resource, source or sink).

A connection point is an abstraction of the physical resources of a switching system connecting legs together.

Legs can have attributes such as bearer capabilities, number of channels, channel direction (e.g., down-link, up-link, bi-directional) type of signalling (e.g. channel associated or common channel signalling), etc.

Attributes of an instance of a connection point are those of/derived from the legs connected to it.

From the feature segment viewpoint a connection point represents the capability of a switching system to bridge information from the legs connected to it. Information from legs can be either information to be broadcast (voice or data) over the joined leg(s) or signaling information (e.g. events). In the latter case the information is bridged according to the rules established for Event Monitoring on legs, on a per call basis.

3.1.2 Primitives on the Connection Control Socket objects

The following are some criteria that justify the choices made for the Connection Control socket primitives:

* Open Interface: the Connection Control operations should be as flexible as possible for future service needs. For this evolution, modifications, especially at the SSF level, must be minimized.

* Service-independent Primitives: this means the service should be as transparent as possible to the SSF. The SCF does not know with which signalling system (No. 7, MF, ..) SSF manages a given call.

  The SSF is responsible for the execution of primitives independently of the exchange type where it is mapped on (local, transit), and of the signalling system required for the specific leg. However, the treatment of a given primitive can be different whether the SSF is mapped on a toll or on a local exchange and even whether the signalling system is N. 7 or MF but it is transparent to the service.

* Service related intelligence in the SCF: normal telephonic intelligence does not need to be duplicated in the SSF and the SCF. The basic call processing is assumed to be already present in the SSF. The SCF needs to cooperate with this basic call processing. Therefore, it needs to have the possibility to treat or to be informed about signalling events received by the SSF.

3.2 Data Management Socket

The Data Management Socket is intended for transfer, between SSF and SCF of data or information that is not necessary for the control of the connection of any individual call. However such information can be related to some IN services or some instances of IN service.

This socket will be mainly used by SCF to retrieve or update existing information residing in the SSF or to ask SSF to collect non signalling events.

Data or information needed for a call or on a per call basis must not be manipulated through this socket but through the connection control socket.

Nevertheless, data concerning service triggering, i.e., the Trigger Table, can be updated via a Data Management Socket (e.g. in order to avoid a direct relationship between the SMF and the SSF, the SCF may have to perform some transit functions between those entities: when a new service or a new service version is introduced, the trigger table needs to be updated).

An important point that justifies the need for the Data Management socket is that information may be stored using a variety of database technologies: the Data Management Socket allows the SCF to manipulate network data uniformly without knowledge of the underlying database systems.

3.3 Status Monitoring Socket

This socket type is introduced in the Connection Control Model to allow the SCF to request status monitoring of logical resources (objects), involved in call processing and not situated at the level of the SCF itself. These objects can be situated at the level of the SSF and/or at the level of the switch where the SSF is mapped with or at the level of an SRF.

The status monitoring is limited to the busy/free state changes of the supervised object, which is not necessarily involved in calls or allocated to legs supervised by that SCF.

Status monitoring may be requested for the next busy/free transition or continuously.

An object in the status monitoring socket may contain a single resource or a group of resources.

An object containing a group of resources is busy when all facilities supported by the group are busy and free when at least one facility of the group is free.

While a supervision is taking place, independent primitives may be performed on the supervised objects via other types of sockets with access to the supervised objects.

The execution of these independent primitives shall not be impacted by the status monitoring request.

The possibility to request the next state change and to reserve the object when it becomes free is for further study.

3.4 Traffic Management Socket

3.4.1 Objectives of In Traffic nana~

IN Traffic Management (IN TM), related only to the traffic that requires interaction between ***SSF'S*** and SCF'S, contains the main following objectives:

* Network protection aaai.nst overload. The objective is to limit the number of IN calls in such a way as to prevent or alleviate a state of congestion in one of the following entities:

  - the whole telecommunications network (PSTN, ISDN, etc.), or just a portion of it (e.g. a switching area), as far as it concerns switches that contain SSF

  - one or more exchanges, e.g. one of the exchanges containing the Service Switching Functions

- a specialized node, e.g. an "intelligent node" containing Service Control Functions

- etc.

* Useless-calls limitation. The term "useless" is applied to IN calls that make use of network resources but otherwise generate no revenue or bear no utility (e.g. calls directed to a non-existing special subscriber code). The objective is to limit their number, even though they are not causing congestion in the network.

* Call limitation on customer request. The objective is to limit the number of calls having given characteristics, which are specified by the customer, even though they are not causing congestion. An example can be 800-service, should the called party specifically request that the number of calls presented to destination does not exceed a given limit: calls in exess are blocked and instead of receiving the usual busy tone they may be given a verbal announcement.

The first two objectives are oriented to the network operator as they tend to optimise network resource usage, nevertheless they may result in a better quality of service for the user. The third objective is customer-oriented, and it may be assimilated to a service feature requested by the customer.

These objectives are transparent to the socket model, i.e. they are achieved via identical operations on the objects.

3.4.2 Technical aspects of IN Traffic Management

Two basic assumptions that underlay the modelling of IN Traffic Management are: 1) that IN TM is only concerned with IN-provided services and related calls, and 2) that IN TM should, as far as possible, be service independent.

The first assumption implies that, per each call received in the SSF, the IN TM mechanisms can only be activated after a trigger is hit, because only at that point in time the SSF becomes aware that the call is an IN call. Since, for the reasons given below, all blocked calls should not generate any SSF-SCF interaction, IN TM mechanisms should in fact take place immediately after a trigger is hit.

The second assumption is that IN TM procedures must work in a way that is independent of the service the call is related to. After a trigger is hit, the only information elements, common to all services, which are available at the SSF are those contained in the Trigger Table, i.e. SCF Addressing Info (SAI) and SCF Export Info (SEI). Therefore, the parameters that can characterize an IN call for the purpose of IN TM are: a) the Trigger Check Point, b) the SAI and c) the SEI: they are contained in the "scope"parameter of IN TM filters.

The IN TM mechanisms cannot be fully understood and analyzed unless some considerations are made on the physical architecture of an IN-structured network. The SCF can be implemented in a specialized node (a "Service Control Point", SCP) or else be resident in other physical nodes. The SSF is in principle implemented in a switching exchange (Service Switching Point, SSP). This implies two considerations:

first that an SCP is in principle a focal point where a large number of IN calls are treated, and therefore it can have a fairly global view of the traffic generated by IN-provided services. Therefore, traffic control is best performed from this privileged viewpoint. This is the reason why IN TM mechanisms are provided at all, as differentiated from other traffic control mechanisms provided by telecommunications networks.

second that the SSP is in principle the point in an IN architecture which is closest to the originators of the IN calls. Therefore, in order to minimize network resource usage, the filtering of IN calls must be done in the SSP, and IN calls which are blocked should proceed no further in the network than the SSP itself (i.e., no inquiry should be sent to the SCP for any of these calls).

Another set of considerations that apply to the Traffic Management as it is described in the T.R., is the followings

- an alternative way of filtering IN traffic is that the SCF should receive a inquiry for each IN call in any case, and thereafter decide whether to respond to the SSF with, a "proceed" instruction or with a "block" one. This may be a desirable feature for some services, however it is in fact a normal interaction between SSF and SCF for call treatment, and that is the reason why it is not included in the mechanisms for IN TM.

- the association between an SSF and an SCF may in some cases be confined only to the duration of the interaction for one call, i.e., the SCF has no knowledge of the set of SSF's it can interact with, and only knows the identity of an SSF when the latter launches a inquiry for the provision of a service. This is the case with mobile applications, e.g. considering UPT services spanning over several countries. However, this should not have any influence on the procedures described for IN TM, but possibly on the service logic that controls such procedures.

- In-structured networks are still telecommunications networks, and as such their management should comply with the principles of TMN. It is foreseen that of necessity there will be several interactions between IN and TMN, and that the two concepts, both being architectural concepts, will have to influence each other and at the end be harmonized together. Traffic management is particularly sensitive to the problem of the TMN/IN interaction, as it is a typical OA&M function.

A possible solution that has been proposed for the interaction between IN and TMN, for the sake of IN TM, is the following: in addition in IN TM sockets between SSF and OCF, sockets of such a type may also opened between SSF and an Operation System Function (OSF). It might be useful that in such an IN TM socket between SSF and OSF, all active filters in the SSF be represented via corresponding object instances, regardless of the SCF that requested the activation of each filter. This proposal is not developed in detail.

- the flow of signalling messages generated in IN TM sockets should be limited to prevent additional overload. As the network is protected starting from the SSF level, it may be of great interest to put the SSF at the local office level. This may request that mechanisms defined for IN TM result in as few messages as possible exchanged between SSF and SCF.

The considerations above give a technical background to the choices made for the IN TM socket primitives, as described in a previous section.

3.5 Resource Control Socket

The objectives for which Resource Control mechanisms are introduced are:

Resources can be allocated within the SSP or outside the SSP (e.g. IPs), a certain types of resources can be centralized and other types can be distributed, but the control and management of resources can be centralized in SCF. It makes the availability of resources more effective by allocating them somewhere and using them anywhere. For instance the unavailability of a given resource or resource type in some SSP can be prevented if SCF knows where such a resource is available. Connection to an alternative pool of resources is then possible.

To allow reservation of resources. It foresees the need of a specific resource type for certain services (or calls) before establishing the connection. Thus resources can be reserved in advance and just allocated at connection time.

To allow SCF to operate on logical resources. Some resources are to be used on a per call basis and some others are not associated to any particular call. On the other hand, some services can require specific resources where as others only require the functionality associated to several resources. The service logic may in some cases indicate exactly which type of resources must be used or it may only indicate the function required. This permits flexibility on resource management. for instance, for the cases where the function required can be provided by different device types or the appropriate resource depends on the user line characteristics and/on terminal equipment capabilities unknown by the service logic.

This socket type is introduced in the Connection Control Model to allow the SCF to operate on resources (objects) used for info transfer between addressable entities and the network during the execution of service logic. Info can be transferred from the network to the addressable entity (e.g. announcements) or from the addressable entity towards the network (e.g. digits).

Resources can be allocated with the SSF or resources can be allocated outside the SSF (e.g. SRFs).

The way the physical allocation of the resources are known in the SCF determines the resource control. Following possibilities should be possible:

* The SCF always gives only a resource reference number and the SSF determines where the resource is located in the network, whereafter the SSF performs the communication.

  The advantage of this mechanism is that the SCF does not need to have a resource control, with the result that the resource control socket is not used in this case.

  In other words the resource related communication is exchanged via the Connection Control Service primitives.

* The SCF knows that the resource is allocated with the SSF or not. In the case that the resource is allocated with the SSF the SCF passes a resource reference number and the SSF performs the requested communication. This case is the same as the previous one. If the resource is not allocated with the SSF two cases can be considered:

  - The SCF passes the address towards the SSF and the SSF performs the communication with the SRF to address the resource.

    The advantage of this mechanism is that the SRF needs only to support the communication with the SSF and not with the SCF. In other words in this case there is no need for a resource control socket, because the communication can be performed via the connection control service primitives.

  - The SCF communicates separately with the SRF to address the resource. the SSF is only used to set up a transport connection to the SRF to address the resource.

    The advantage of this mechanism is that resources for which control is passed to the SCF can be used by the SCF as long as the control is not passed back. In other words the SCF may address the same resource, while the resource is connected successively to different legs in the same or different calls, controlled via a connection control socket in the same SCF. This is the only case that requires a resource control socket to address the requested resource.

All the above described possibilities shall be supported and it is up to the SCF to select the most appropriate method to address resources during the execution of the service logic.

4 SWITCHING PERSPECTIVE OF THE CONNECTION CONTROL MODEL

4.1 Introduction

This section presents an alternative approach to the definition of the Connection Control Model. The actual relationship between this approach and the one described in the previous sections is not contained in this T.R. and may be the subject of future studies.

The principle of the Call Control is based on the idea to separate the call control into two parts:

SERVICE CONTROL

and

CONNECTION CONTROL

The Connection Control will in principle be controlled by the Service Control, but might also be controlled by another Connection Control. This is implicated in the fact that a connection is controlled by two or more Connection Controls, each of them responsible for a certain part of a connection. A connection is established between two or more parties.

It is decided that the long term studies of IN should leave out terms like local exchanges, transit exchanges etc. The conceptual model, the starting point of the top down study of ETSI and CCITT on IN, has replaced these terms with a more general term, Physical Entity. The reason is that the character of the network might change radically after the introduction of IN. Existing terms will confuse the picture of future IN's. It should, however, not lead to serious constraints to IN studies to retain the connection as an important ingredient also in future networks. Connections will be unavoidable and the treatment of connections is an experience that should be saved from existing networks over to future networks. The treatment will certainly need to be better and more detailed described end above all adapted to IN ideas.

4.2 The bottom up basis for the IN studies

A basic idea of Connection Control is to distribute control to all Physical Entities providing elements of a connection. The term Exchange Connection used in the past shall be replaced by the term Physical Entity Connection. Future connections will as in the past, consist of one or more concatenated Physical Entity Connections.

The study of the Connection shall be the starting point for the study of IN from a bottom up point of view. The study will start with a Connection which is familiar and well known, the Connection of a Basic Call. The characteristics of this Connection shall be identified and highlighted. All other Connections shall be considered as deviations from the Connection of a Basic Call. This Connection will be named Basic Connection.

The Basic Connection shall be given characteristics which will make it suitable for application in an Intelligent Network (IN). It will be necessary to study the Basic Connection in detail to harmonize the possibilities to influence the Connections and to describe Services. It is the intention to describe the Basic Connection and its characteristics in the physical plane and then to transfer the study to the functional plane.

4.3 Physical Entity Connections

Recommendation CCITT Q.522 defines following Physical Entity Connections: Internal Connection (IC), Originating Connection (OR), Terminating Connection (TE) and Transit Connection (TR).

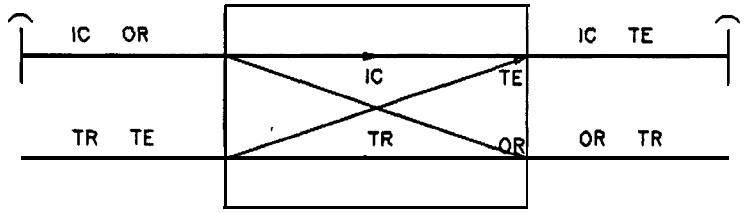


Fig 1

Fig. 1 shows a Physical Entity with these Physical Entity Connections. A subscriber might establish originating Connections and Internal Connections. He might receive Terminating Connections and Internal Connections. The Physical Entity might receive from the network Terminating Connections and Transit Connections.

4.4 Control Fields and Transition Points

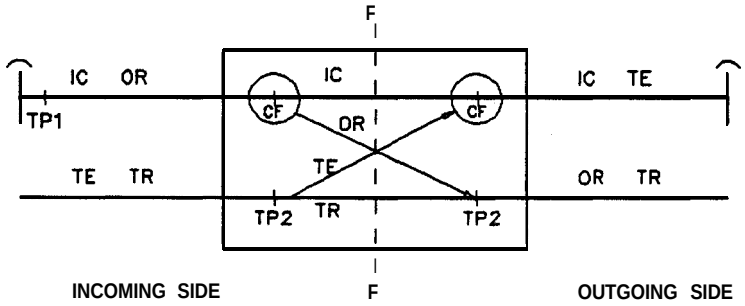


Fig 2

Fig 2 shows the same Physical Entity Connections as in Fig. 1 but additional symbols have been introduced.

A Transition Point 1 (TP1) has been introduced to identify the area of resource control responsibility. The SPN (Subscriber Premises Network) is responsible for the Connection on the subscriber side of Transition Point 1 (TP1) and the Connection Control is responsible on the network side of Transition Point 1 (TP1). The exact location of Transition Point 1 is of no interest to the Connection Control. Transition Point 1 (TP1) data of interest to Connection Control are its identification and characteristics. This set of characteristics shall be defined as a subset of the subscriber route characteristics.

It is an important aspect of IN to discuss the characteristics as possible variables and to take notice of the variability in the detailed description of the Connection Control. Different types of subscriber lines might appear behind the Transition Point 1 (e.g. satellite links, radio links for different mobile services, normal telephone lines etc).

A Control Field (CF Fig 2) symbolized by means of a circle represents all functions of a Physical Entity which might process or influence the information flowing through a Connection. The functions might in real implementations be distributed throughout the Physical Entity.The logic of Connection Control will always consider the functions as allocated to the Control Field (CF). The complications that a distribution of functions might introduce is a problem the manufacturer will have to solve and will not be a problem to consider by Connection Control.

A Transition Point 2 (TP2) identifies an inlet or an outlet of an incoming or outgoing trunk of a Physical Entity Connection. Data of Transition Point 2 (TP2) of interest to a Physical Entity Connection Control are identification and characteristics. Characteristics are given by incoming or outgoing route data.

4.5 Subscriber Legs, Internal Paths, Incoming– and Outgoing Trunk Legs

A Physical Entity Connection will always be divided into three parts. These three parts will be given an individual treatment by Connection Control. Each part is selected out of four types which are defined as Subscriber Leg, Outgoing Trunk Leg,Incoming Trunk Leg and Internal Path. Two or three of these parts will be present in every Physical Entity Connection. Two parts might be of the same type.

Definition of a Subscriber Leg is:

A Subscriber Leg of a Physical Entity is a connection element reaching from and including a Control Field of the Physical Entity to a Transition Point 1 (TP1). (Fig 2).

Definition of an Incoming Trunk Leg is:

An Incoming Trunk Leg of a Physical Entity is a connection element reaching from a Transition point 2 (TP2) of a preceding Physical Entity to a Transition Point 2 (TP2) of the Physical Entity. (Fig 2).

Definition of an Outgoing Trunk Leg is:

An Outgoing Trunk Leg of a Physical Entity is a connection element reaching from a Transition Point 2 (TP2) of the Physical Entity to a Transition Point 2 (TP2) of a next succeeding Physical Entity.

Definition of an Internal Path:

An Internal Path is a connection element interconnecting a Subscriber Leg of a Physical Entity with an Outgoing Trunk Leg of the same Physical Entity or interconnecting an Incoming Subscriber Leg of a Physical Entity with a Subscriber Leg of the same Physical Entity or interconnecting a Subscriber Leg of a Physical Entity with another Subscriber Leg of the same Physical Entity or interconnecting an Incoming Trunk Leg of Physical Entity with an Outgoing Trunk Leg of the same Physical Entity.

Internal Physical Entity Connection:

An Internal Physical Entity Connection will consist of a Subscriber Leg of a calling subscriber, a Subscriber Leg of a called subscriber and an Internal Path interconnecting the two Subscriber Legs.

Originating Physical Entity Connection:

An Originating Physical Entity Connection will consist of a Subscriber Leg of a calling subscriber, an Outgoing Trunk Leg and an Internal Path interconnecting the Subscriber Leg and the Outgoing Trunk Leg.

Terminating Physical Entity Connection:

A Terminating Physical Entity Connection will consist of an Incoming Trunk Leg, a Subscriber Leg and an Internal Path interconnecting the Incoming Trunk Leg and the Subscriber Leg.

Transit Physical Entity Connection:

A Transit Physical Entity Connection will consist of an Incoming Trunk Leg, an Outgoing Trunk Leg and an Internal Path interconnecting the Incoming Trunk Leg and the Outgoing Trunk Leg.

4.6 Control Field functions related to Connection Control

A Control Field will according to its definition posses all functions of a Physical Entity capable to process or influence an information;, flow through a connection of which the Control Field is a part.

Only a limited number of these functions are controlled by Connection Control, but these set of functions is available to all services applying Connection Control. The rest of the functions will be directly controlled by Service Control itself and they will be unknown to Connection Control

The set of functions accessible by Connection Control is:

Switch Through Connection Function
Split Connection Function
Join connection Function
Seize Control Field Function
Release Control Field Function
Connect Transition Point 1 Function
Disconnect Transition Point 1 Function
Connect Internal Path Function
Disconnect Internal Path Function

Switch Through Connection Function

Switch through connection is the traditional switch through of a connection at the moment the connection is established and a conversation might start as soon as the connection is switched through.

Split Connection Function

A split connection function will be activated at an initiative of a subscriber who for some reasons wants to disconnect an established connection but at the same time does not want to release it.

Join Connection Function

A join connection function will be activated at an initiative of a subscriber who wants to reestablish a connection which was splitted. A join connection function will also be activated to connect a Subscriber Leg with an Internal Path belonging to another connection.

Seize Control Field Function

A seize Control Field function is activated during an establishment of a Subscriber Leg. It seems to be reasonable not only to activate this function as a part of another function but to have a possibility to activate the function as an independent function.

Release Control Field Function

A Release Control Field Function is activated during a Release of a Subscriber Leg. It seems to be reasonable to have a possibility to activate this function as an independent function.

Connect Transition Point 1 Function

A Connect Transition Point 1 function is activated during an establishment of a Subscriber Leg. A Connect Transition Point 1 function interconnects a Control Field and a connection element between a Control Field and a Transition point 1. A connection element might include concentrators, switching stages, multiplexer and other network entities.

Disconnect Transition Point 1 Function

A Disconnect Transition Point 1 Function is activated during a release of a Subscriber Leg. It disconnects a Control Field from a connection element to which a Control Field is connected.

Connect Internal Path Function

A Connect Internal Path Function is a function which will inter connect a Control Field with an Internal Path. This function will not exist as an independent function, but as a part of other functions.

Disconnect Internal Path Function

A Disconnect Internal Path Function is a function which will disconnect a Control Field from an Internal Path. This function will not exist as an independent function, but as a part of other functions.

4.7 Applied symbols of Physical Entity Connections

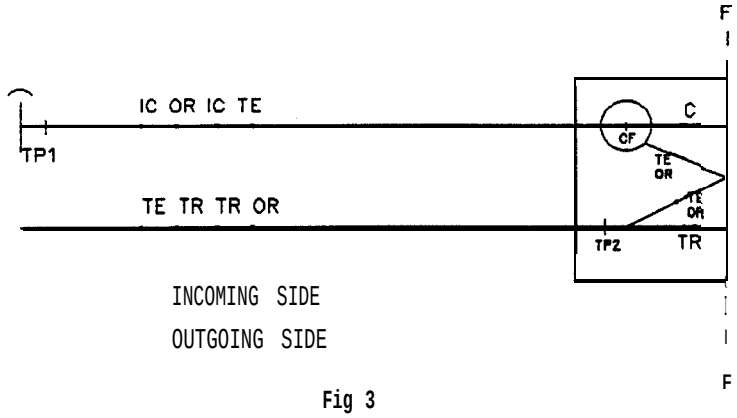


Fig 3

Fig 3 is identical to fig *2* but is the folded version. The connection diagram is folded along a line FF of fig 2. A FF-line is called a reflection line. Fig 3 shows only the connection principle of a Physical Entity in a symbolic manner. It shows Subscriber Leg, Outgoing Trunk Leg, Incoming Trunk Leg and Internal Path. It identifies symbols to be applied in drawings.

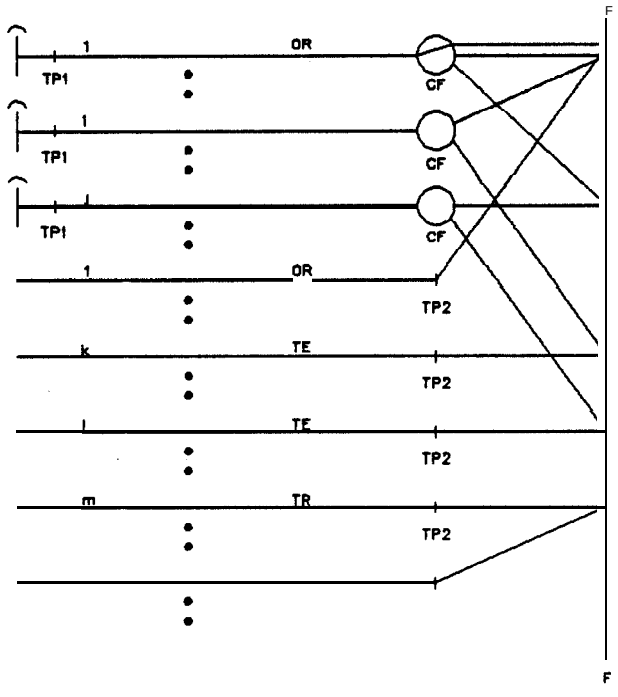


Fig 4

Fig 4 shows an application of symbols introduced in fig 3. A main difference of fig 4 compared to fig 3 is that every active Physical Entity Connection is presented in fig 4. Fig 3 shows only in a symbolic manner Subscriber Leg, Outgoing Trunk Leg and Incoming Trunk Leg. Fig 4 shows each individual Subscriber Leg, Outgoing Trunk Leg and Incoming Trunk Leg.

Subscriber Legs of fig 4 are given numbers from 1 to j. Subscriber Legs l,i and j are active. Incoming Trunk Legs and Outgoing Trunk Legs are numbered from 1 to n. Outgoing Trunk Legs indicated as active in fig 4 have the numbers 1 and n. Incoming Trunk Legs also indicated as active have the numbers k,l and m.

It is made possible to see the difference between Incoming Trunk Legs and Outgoing Trunk Legs. An Internal Path end interconnected with an Incoming Trunk Leg is drawn perpendicular to the Reflection Line. The same rule is also valid for Subscriber Legs. An Internal Path end connected to a Control Field of an Originating Connection is drawn perpendicular to the Reflection Line.

4.8 An initial discussion of connections

Subscriber Leg 1 (fig 4) is a part of an Originating Connection. The Internal Path end of Subscriber Leg 1 connected to the Control Field is drawn perpendicular to the Reflection Line. The connection is an Originating Connection leaving the Physical Entity as Outgoing Trunk Leg 1. A full line drawn across the Control Field shows that Subscriber Leg 1 is connected to Outgoing Trunk Leg 1.

The subscriber controlling Subscriber Leg 1 has also established a second connection, an Internal Connection, to a subscriber controlling Subscriber Leg i. The connection, an Internal Connection, is splitted and waiting to be connected to Subscriber Leg 1.

A third connection, also an Internal Connection is splitted. This connection is also initiated by the subscriber controlling Subscriber Leg 1.

A subscriber controlling Subscriber Leg 1 is involved in three calls at the same time. He has initiated establishment of two Originating Connections and has permitted a third connection to be connected to his Subscriber Leg.

A subscriber controlling Subscriber Leg i has accepted reception of two connections and will have possibilities to switch between the two connections.

A subscriber controlling Subscriber Leg j has established one Originating Connection and has accepted a second connection to be connected to his Subscriber Leg. He will then be able to switch between one Originating Connection and one Terminating Connection. It shall be noticed that the Subscriber Leg is at the same time applied for Originating Connections and for Terminating Connections.

4.9 An initial discussion of Connection Control and Service Control

Different connections do not need to be interrelated. They might be initiated from different Service Functions even if they apply to the same Subscriber Leg. It is therefore necessary to get a closer look at resource control. It is a must to keep flexibility , but to limit complexity of Connection Control. According to ideas of IN, Service Functions might be allocated to different physical entities throughout a network. It seems therefore reasonable to allocate resource control to physical entities providing resources. A resource control or Connection Control should be separated from a Service Control.

4.10 Some characteristics of Connection Control and Service control

Some characteristics of Connection Control and Service Control will have to be identified.

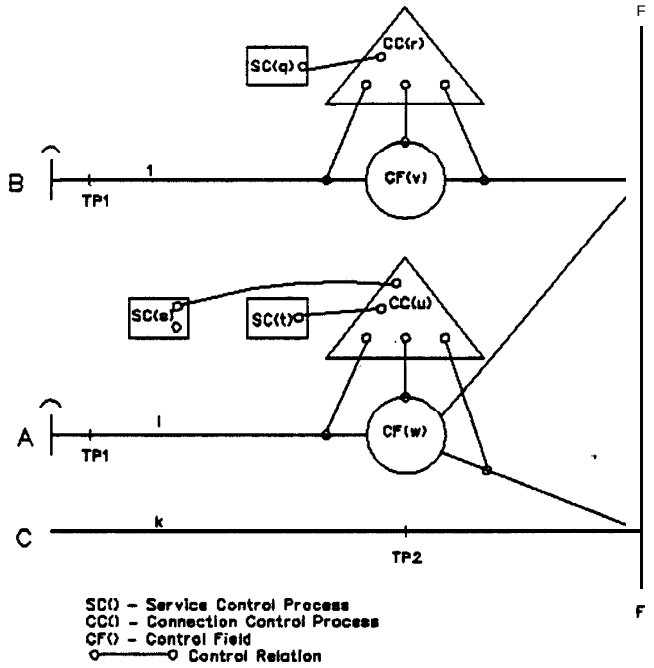


Fig 5

Fig 5 shows an excerpt of fig 4. Established connections are somewhat simplified compared to connections of fig 4. Subscriber B controlling Subscriber Leg 1 has only established one connection (Internal Connection) . Subscriber A controlling Subscriber Leg i has accepted establishment of two connections. Both connections are going to share Subscriber Leg i. The connections are one Terminating Connection and one Internal Connection.

Compared to fig 3 and fig 4 two additional symbols a triangle and a quadrangle have been introduced. A triangle is a symbol of a Physical Entity Connection Control dealing with establishment of connections. A Physical Entity Connection Control process is not only dealing with establishment of connections and parts of connections (Legs, Internal Paths and Trunks) but must also be considered as a resource manager during the time resources are activated for a connection controlled by a Physical Entity Connection Control process.

A quadrangle is a symbol of a Service Control. A Service Control might be located to the same Physical Entity as a Physical Entity Connection Control, but might also be located somewhere else in the network. A Connection Control will establish a connection at a request of a Service Control or at a request of another Connection Control.

A connection controlled by Subscriber B is established by Physical Entity Connection Control CC(r) (fig 5). A Physical Entity Connection is established in three steps.

A first step is to establish Subscriber Leg 1. Subscriber Leg 1 is including Control Field CF(v). A second step is to receive the identification of Control Field CF(w). Control Field CF(w) belongs to Subscriber Leg i which is established and supervised by Physical Entity Connection Control process CC(u). A third step is to establish an Internal Path between Control Field CF(v) and Control Field CF(w). Physical Entity Connection Control process will establish Subscriber Leg 1 and will also establish an Internal Path between the two Control Fields CF(v) and CF(w). Subscriber Leg i, however, is established by Physical Entity Connection Control process CC(u). This process is directly under control of subscriber A (fig 5). Additionally it shall be noticed that subscriber A also has a second connection connected to Subscriber Leg i. A Terminating Connection using Incoming Trunk Leg k is connected to Control Field CF(w) (fig 5). Physical Entity Connection Control process CC(u) has established Subscriber Leg i including Control Field CF(w). The process CC(u) has first established Incoming Trunk Leg k and finally established an Internal Path between Control Field CF(w) and Transition Point TP2.

Physical Entity Connection Control CC(u) is controlling and supervising all resources related to an application of Subscriber Leg i. A Symbol, a lever, is indicating what process is controlling a certain resource.

Two Service Control processes are involved to have their connections established. They will both have to share Subscriber Leg i. Subscriber A has required this solution. Service Control process SC(s) interworks with Physical Entity Connection Control process CC(u). If Subscriber B has established his connection to subscriber A after A has received the Terminating Connection then Subscriber Leg i will already have been established. Service Control process SC(s) will only have to require Join/Split functions from Physical Entity Connection Control process. Service Control process SC(t) is the service requesting a Terminating Connection to A. The two Service Control processes SC(s) and SC(t) are totally independent of each other. The two Service Control processes SC(s) and SC(q) are two identical processes. They interwork to provide A and B with the requested service. Service Control processes might be colocated with Physical Entity Connection Control processes or might not be colocated. A mixture of locations shall also be possible.

4.11 Connections

As stated in an introduction to this chapter connections will be established by concatenating Physical Entity connections. A Connection will in principle consist of one Originating Physical Entity Connection concatenated with Physical Entity Transit Connections and at the end of the connection have a Terminating Physical Entity Connection.

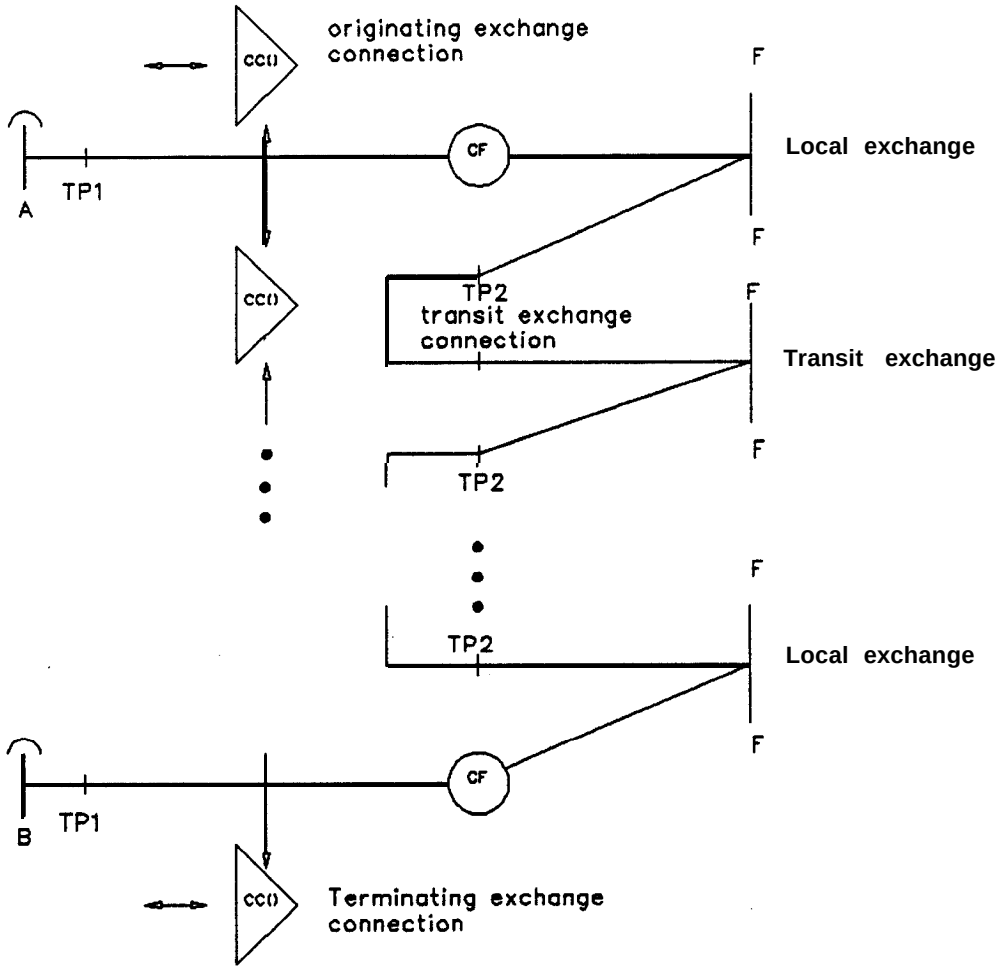


Fig 6.

It should be noted that only a Originating Physical Entity Connection and a Terminating Physical Entity Connection will have Control Fields. One for each party of a connection. Fig 6 shows an example of this connection. Fig 7a shows an Internal Physical Entity Connection and 7b a connection consisting of an Originating Physical Entity Connection and a Terminating Physical Entity Connection.

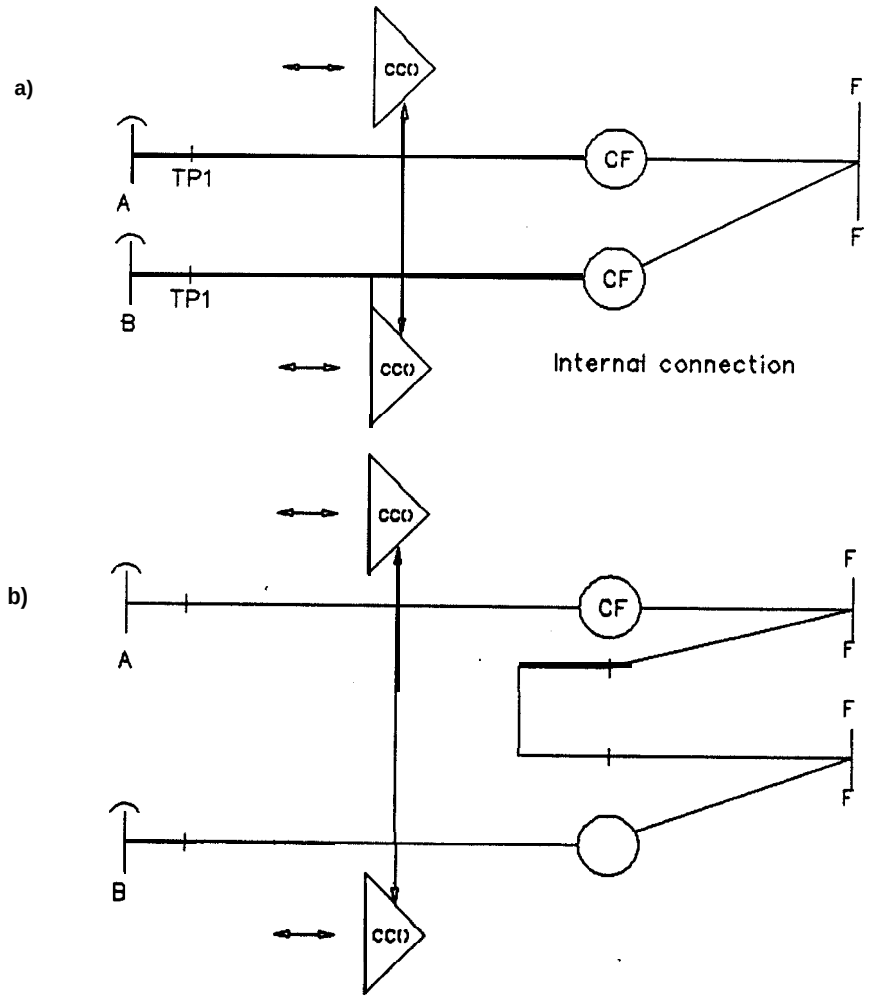


Fig 7.

4.12 A Bidirectional symmetrical Basic Call Model

As stated in an introduction to this chapter a basic call shall be a starting point of bottom up studies of IN. It is from a traditional point of view and also from an experimental point of view reasonable to give a Basic Call Model certain fundamental characteristics.

A Basic Call Model shall have a connection established between two parties

A connection shall be bidirectional

Information type exchanged between the two parties shall be the same in both directions (e.g. sound).

Both parties shall have the same possibilities to give instructions to the network and to receive information from the network.

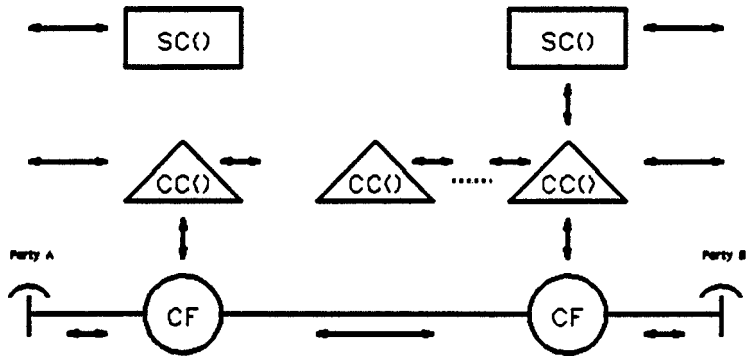
Each party is served by its own Service Control process. Both processes are assumed to be identical.

Each party controls its own Control Field.

Only one information type is transferred by the connection. The information type is specified in the I-series of CCITT Recs.

A Control Field will only take part in one Subscriber Leg.

A limited number of Internal Paths might be connected to a Control Field of a Basic Call.



- Bidirectional Identical Information flow
- Two identical Control Fields
- Two identical Service Control processes
- Two identical Connection Control processes

Fig.8 A Bidirectional Symmetric Basic Model

Fig 8 shows a Basic Call Model. Both parties have established a relationship to a Service Control process. The two processes are identical. Each Service Control process has established a relationship to a Physical Entity Connection Control process. Both Physical Entity Connection Control processes and both Control Fields are identical. It is assumed that each SPN (Suscriber Premises Network) has a connection control function and a independent service control function.

Each process will only react on instructions from the party which has created them. They will, however, interwork with the service control process of the other party as a consequence of instruction from its own party. A Service Control process might be colocated with a Control Field of the Subscriber Leg or might be located to any Physical Entity which will communicate appropriately with the other process involved in the Basic Call.

A Basic Call will at least have two Physical Entity Connection Control processes. Even if both parties, taking part in the Basic Call, are connected to the same Physical Entity, a Physical Entity Connection Control process will control maximum one Subscriber Leg.

A Basic Call Model assumes that each Service Control Process is only communicating with one Physical Entity Connection Control process. This Physical Entity Connection Control process is controlling only one Control Field. A Control Field of a Basic Call Model provides five functions:

Connection of Subscriber Leg to a Control Field

Connection of one/several Internal Paths to Control Field

Switch through a connection

Split a connection

Join a connection

A Control Field is by means of these functions capable of influencing the information flow through a Physical Entity. A Control Field might however, process information according to special service requirements. A Physical Entity Connection Control is not familiar with these functions, but will have to be able to differentiate between different Control Field types. A Service Control will Then have direct access to Control Fields, but only for functions not related to functions controlled by a Physical Entity Connection Control.

The Physical Entity Connection Control for Basic Connections is identical for all Physical Entities providing Physical Entity Connection Control. Even for Physical Entities only providing Transit Connections a Physical Entity Connection Control will be the same as for a Physical Entity providing Transit, Terminating, Originating and Internal Connections.

4.13 Symmetrical Extended Basic Call

A Basic Call model will be extended. An extension of the model will be appropriate for connections requiring special information processing capabilities. An example is a conference bridge. Generally a Basic Call model will be extended with a Control Field. If a Basic Call model is studied a symmetri will require that an extra Control Field is introduced for each party taking part in a call.

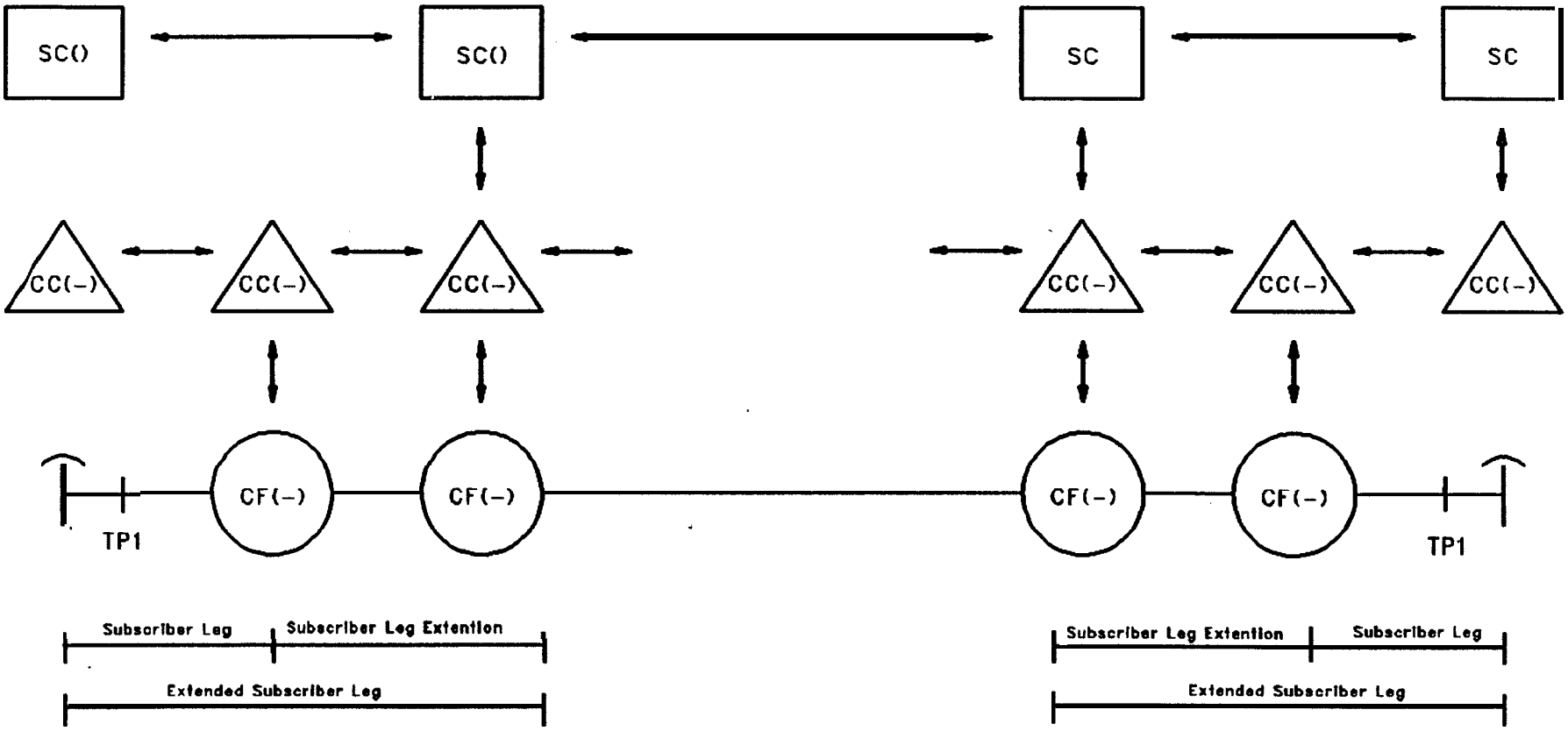


Fig. 9 Symmetrical Extended Basic Call Model

Fig 9 shows an Extended Basic Call Model. The model presents an Extended Subscriber Leg as a concatenation of a Subscriber Leg identical to a Subscriber Leg of a Basic Connection and a Subscriber Leg Extension. A Subscriber Leg Extension will at least consist of an Internal Path and a Control Field, but might additionally include Outgoing Trunk Legs and Incoming Trunk Legs. An Extended Basic Call Model is symmetrical.

4.14 Asymmetrical Extended Basic Call Model

Fig 10 shows an Asymmetrical Extended Basic Call Model. One party is connected to an Extended Subscriber Leg. It is natural to imagine that this party might be the one taking an initiative to establish the service. A Control Field of a Subscriber Leg Extension will have the same capabilities as a Control Field of a Basic Connection but will additionally have capabilities required for a certain service.

A second party or all other parties involved are connected by means of normal Subscriber Legs. Fig 10 indicates a Control Field with capabilities as a Control Field of a Basic Connection with a B. A Control Field with additional information processing capabilities is indicated by means of an S (Specialized).

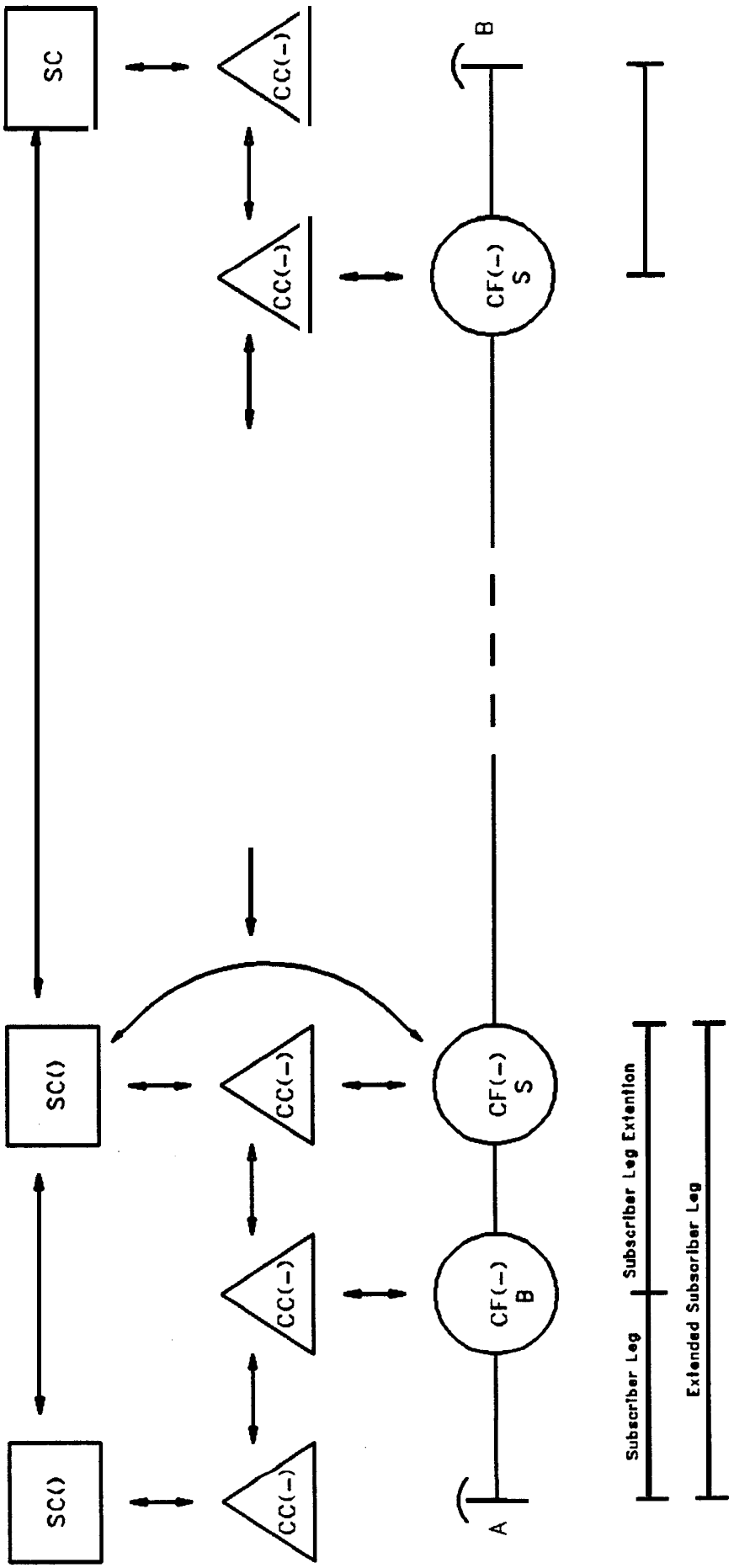


Fig. 10 Asymmetrical Extended Basic Call Model

ANNEX 1: FORMAL SPECIFICATION OF CCM PRIMITIVES

Proposed methodologies for the formal specification of primitives and parameters described in previous section, are the following:

- ASN.1

- Object Oriented techniques

The following text is an example of a formal specification of connection control socket primitives, which is not complete, but which gives a more formal description of the text included in section 2.1.

```
CC-PROVIDE-INSTRUCTION PARAMETER : = SEQUENCE { TriggerCondition,
SEQUENCE OF { CHOICE { CP, LEG }}}

CC-CREATE PARAMETER : = SEQUENCE { LEG, AddressableEntity, OtherLeg,
CompletionCondition }

CC-FREE PARAMETER : = SEQUENCE { LegId, ReleaseCause, When }

CC-JOIN PARAMETER : = CHOICE { SEQUENCE { LegId, LegId, CP },
                               SEQUENCE { LegId, CPId },
                               SEQUENCE { CPId, CPId }}

CC-SPLIT PARAMETER : = CHOICE { CPId, LegId }

CC-MONITOR PARAMETER : = SEQUENCE { LegId, SEQUENCE OF { EventId,
EventMode }, OtherLeg }

CC-GENERATE PARAMETER : = SEQUENCE { LegId, EventId, EventParameters }

CC-EVENT PARAMETER : = SEQUENCE { LegId, EventId, EventParameters;
EventDate }

CC-SEND-RECEIVE PARAMETER : = SEQUENCE { LegId, InfoToSend,
InfoToReceive }

CC-SEND-RECEIVE RETURN PARAMETER : = InfoReceived

CC-RESUME PARAMETER : = SEQUENCE OF { CPId }

CC-DETACH PARAMETER : = SEQUENCE { LegId, AbsoluteReference }

CC-ATTACH PARAMETER : = SEQUENCE { LegId, AbsoluteReference }

CC-MODIFY PARAMETER : = SEQUENCE { LegId, LegAttributes }

CC-SEND PARAMETER : = SEQUENCE { COId, InfoToSend }

LEG: = SEQUENCE { LegId, LegAttributes }

CP: = SEQUENCE { CPId, CPAttributes, SEQUENCE OF { LegId }}

LegId: = INTEGER
          <to identify a leg uniquely within a socket for SSF and SCF>

CPId: = INTEGER
          < to identify a CP uniquely within a socket for SSF and SCF>
```

```
CPAttributes:  = CPmaxNumberOfLeg INTEGER

LegAttributes:  = SEQUENCE { CalledPartyNumber, CallingPartyNumber,
CallingPartyCategory, UserServiceInformation, CUGInterlockCode,
RedirectionInformation, RedirectingNumber, OriginalCalledNumber }
(ISUP parameters)

<to indicate detailed characteristics of a leg>

AddressableEntity: = CalledPartyNumber (ISDN parameter)
<to identify an addressable entity>

TriggerCondition: = ANY
<to identify the trigger event and condition>

OtherLeg: = LegId
<to identify another leg>

CompletionCondition: = CHOICE { OnAnswer, OnAlert, OnCallSent }
<to indicate, when the CC-CREATE can be considered as completed>
<by the SSF and when the next operation on this leg can start>

ReleaseCause: = ANY
<to identify the release cause (similar to ISUP CauseIndicators)>

EventId: = ANY
<to identify an event (signalling system independent)>

EventMode: = CHOICE { Transparent, Intercepted, Duplicated, Ignored }
<to define in the event processing mode>

EventParameters: = ANY
<to identify an event (signalling system independent)>

InfoToSend: = ANY
<to identify what must be sent or broadcast>

InfoToReceive: = ANY
<to identify what is expected to be received>

InfoReceived: = ANY
<to identify what was received>

EventDate: = ANY
<to date an event>

When: = CHOICE < Immediately, AfterCompletionOfOtherOperations }
<to state that an operation must be handled immediately>
<any other operation on this leg being stopped>

AbsoluteReference:  = ANY
<to identify an absolute (network wide) reference>
```

Part 2: The Service Switching Functions (SSF)

Blank page

1 INTRODUCTION

1.1 Rationale

This section provides an abstraction of IN service/feature processing of the IN Functional Entities identified in the IN Distribution Functional plane by the tuple SSF/CCF (Service Switching and Call Control Functions), and later on referred to as SSF when there is no need to characterize specific CCF capabilities.

This abstraction is required

- to provide an observable view of SSF/CCF activities and resources to the other IN Functional Entities, through which the IN FEs can interact with the SSF in the course of executing service logic. This observable view is later developed into physical interfaces and protocol stacks in part III.

- by IN architects as a tool to model a call and to understand and describe the distribution of functions between FES, and FE relationships. These functions are reusable, in that the same function can be reused for a variety of services

- to provide a framework for ensuring correct sequencing of functions within the SSF, and rules for representing and handling service/feature interactions

Thus it must first be high level, and service/vendor implementation independent.

1.2 SSF Domains

Functional Entities identified in the IN DFP which have interaction with SSF/SCF are pictured in figure 1.2.1 and are described below:

SSF The service switching function corresponds to the call processing functionality resident in the switching node of the network, and is primarily for the provision of connection (i.e. communication paths) between the end-users. The SSF interfaces with the connection control functional entity (CCF) and the service control functional entity (SCF). It allows CCF to be directed by the SCF. The grouping of functions in the SSF is for the concern of separating connection control function from the service logic.

CCF The connection control function includes the functionality needed for the execution of a basic call. The CCF interfaces with the SSF, the call control access functional entity (CCAF) and the specialised resource functional entity (SRF). The grouping of functions in the CCF is for the concern of separating the basic call functions from IN related capabilities.

CCAF The call control access function represents the access point of a logical transport path into a switching node (e.g. subscriber line, trunk circuit). The grouping of functions in the CCAF separates the concern of managing a particular user interface from other functional entities, and provide service access independence of the underlying technologies. The CCAF interfaces with the CCF.

Functions in the SSF, CCF and CCAF are accessible from other functional entities and users.

SRF The SRF may be implemented at different locations in the network i.e. in different physical entities (PE), either embedded in existing network resources (e.g. switches) or separately in Intelligent Peripheral connected to the network.

Thus, its use might depend on the state of the transmission path through which it ,is connected.

The SRF may need specific SRF data.

The SRF groups the set of resources used:

- to receive information from,

- to send information to and

- to convert information for the users.

The rationale of this grouping in a particular FE is the provision of complementary capabilities for which existing network FE's might not be equipped while ensuring separation of service logic from technology dependent concerns.

SCF The Service Control Function (SCF) is a functional entity which provides the functionality for the control of IN service processi-on activity. This functionality contains IN Service Logic Programs which are created by the Service Creation Environment Function (SCEF) and managed via the Service Management Function (SMF). This functionality also contains some data related to the SCF.

It has interfaces to the following Functional Entities (FEs):

- SMF - for service management,
- SDF - for specialised data base,
- SRF - for specialised resources, and
- SSF - for service switching capabilities

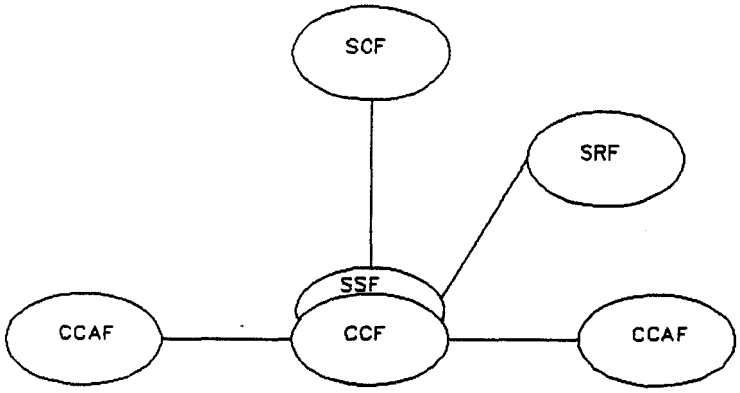


Fig. 1.2.1: FEs interacting with SSF/CCF

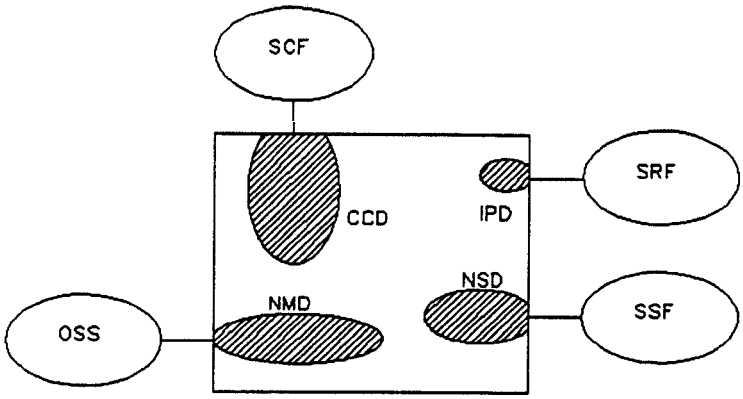
SSF- Service Switching Functions
CCF- Call Control Function
CCAF- Service Control Function
SRF- Specialized Resource Function

The interaction with CCAF, by objective of the IN architecture, is not affected by IN. Thus this interaction is outside of the scope of this section. Other interactions not represented in fig 1.2.1 need also to be considered, as they indirectly relate to IN:

- SSF to SSF to handle service/feature interaction

- SSF to an Operational Support System to handle IN network management requests

Thus the SSF should address 4 domains, pictured below:



Fig, 1.22: SSF domains

CCD - Connection Control Domain
SCF - Service Control Function
OSS - Operational Support System
SRF - Specialized Resource Function
SSF - Service Switching Function
IPD - Intelligent Peripheral Domain
NSD - Network Signalling Domain
NMD - Network Management Domain

These domains are referred to as

- Connection Control Domain (CCD) which interacts with SCF. It correspond to the term "Call Modelling" in CCITT (6.2 and "Connection Control Model" in ETSI NA6)

- Intelligent Peripheral Domain (IPD) which interacts with SRF

- Network signalling Domain (NSD) which interacts with a remote SSF

- Network Management Domain (NMD) which interacts with Operational Support Systems

Each domain will be described independently of each other in chapters 4 through 7 respectively. Chapter 2 explains the rules used in this document to specify each domain. Chapter 3 describes interactions (between domains).

There may be a need to further define subdomains for the purpose of relating part II of the document with each Socket type identified in part I chapter 2 for the Connection Control Model (which only addresses the SCF to SSF interaction).

The need to use a socket type of mechanisms for other interactions has not been demonstrated.

1.3 Processing capabilities

While domains and subdomains represent "subjects" of concern to SSF which need to be adressed for the purpose of modelling the interactions with other functional entities, they should not be interpreted as representing the actual structure of the physical entity hosting the SSF.

On the other hand the bottom-up approach favoured by SPS3 implies that special attention be given to consideration on feasibility, performances and migration from existing SSF implementations. Such criteria are described in chapter 2.1 (SSF DOMAIN specification criteria).

This implies that the internal structure of the SSF should be represented (or modelled) in a way which allows to validate that the specification criteria are correctly met.

Furthermore, some representation of the internal structure is required to provide switch vendors with framework and rules to validate their own implementation.

This document provides such representation of the internal structure through the concept of "processing capabilities", generally named "manager". They don't represent the actual structure of the switch but rather a logical abstraction of what capabilities may need to be added or modified to support the IN architecture.

Each processing capability/manager is described for each domain in terms of

- high level description of what it does
- a model to picture information flow and
- relationships with other processing capabilities
- what is made externally visible
- how this relates and maps with sockets and objects on the external interface
- specifications and constraints to develop operations and events for each relevant object

The symbols to be used to represent processing capabilities (manager) and visibility over the domains/subdomains are explained in the following picture:

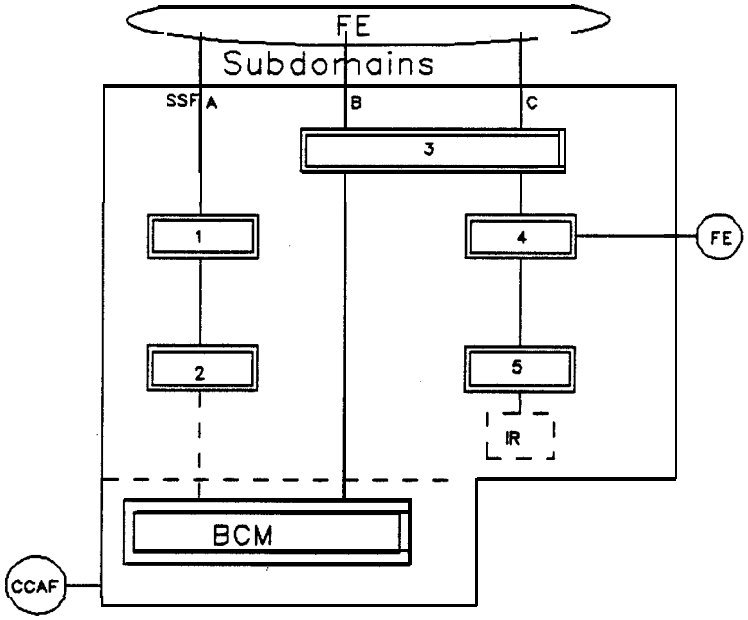


Fig. 1.3.1. Representatlon of processing capabilities

Note: BCM stands for Basic Connection Manager
PC-Proceeding Capability

In this example:
PC1 and PC2 are specific to subdomain A
PC3 is shared between subdomain B and C
PC4 is shared between domains
PC5 does not use BCM but access internal resource (e.g. a file)

- a solid line means that visibility of this processing capabilities is required from the FE

- a dotted line means that visibility is not required but that the relationship needs to be described to assess feasibility

2 RULES FOR THE SPECIFICATION OF SSF DOMAIMS

2.1 SSF DOMAIN specification criteria

Some of the general criteria are given below. The SSF domain specifications should:

a) provide a high-level, vendor implementation independent abstraction of call/service processing functions that implies

- a generic call model supporting all user access technologies under consideration (e.g., POTS, ISDN, etc.), and

- uniformity of functions across multiple vendor products,

b) present an observable view of a Service Switching Function to an IN Service Control Function

c) take into account the existing base of evolvable network technology and the longer term need in its continuing evolution by providing an overall IN call/service processing structure from which useable, coherent subsets of its capabilities, as well as optional capabilities, can be defined as appropriate for a given capability set. F.F.S.

d) provide a framework for defining the information flows (relationships)between a Service Switching Function and an IN Service Control Function, without any assumptions about the physical implementation or distribution of functions

- this implies the need to support one or more concurrent instances of the IN Service Control Function interacting with a Service Switching Function on a single call/service (given item g. below),

e) provide a framework for defining triggering requirements

f) provide a framework for ensuring correct sequencing of functions within a Service Switching Function, and

g) provide rules of representing and handling services/features interactions to support

- multiple concurrent instances of IN services/features on a single call

- concurrent instances of IN services/features and non-IN services/features (e.g., existing switch-based features) on a single call

2.2 SSF DOMAIN representation/modelling rules

An "FSM model" is an operational (dynamic) model of a system that is described by the finite set of states the system can be in and the finite set of transitions possible from one state to another state. Specific operations or activities being performed by the system at a given point in time are associated with specific states. Specific inputs to or outputs from a given state (referred to as "events") are associated with each transition from one state to another state. For a given event, the transition form one state to another will always be the same (i.e., the FSM model is deterministic).

To describe a system in terms of an FSM model, it is first necessary to characterize the state of the system. Objectoriented techniques provide a means to do so by first characterizing the system as a set of one or more "objects" that represent the system properties of interest (e.g., a SM or socket contains objects that represent the connectivity properties of the SSF that are of interest to an SCF). These abstract objects are each described in terms of the unique characteristics they represent (referred to as "attributes") and the actions that can manipulate those objects (referred to as "functions"). As a result, the objects are modular, autonomous entities that can be freely combined and reused (e.g., SM or socket objects provide a robust set of call processing capabilities that can be freely combined and reused by an SCF). In addition, their description can be at an appropriate level of abstraction to hide the technical complexity, physical details, and evolution of the underlying technology of the system.

Given the set of objects that characterize the system (e.g. legs and connection points), the state of the system and its operation can be described. For a given instance of an object, its attributes have specific "values" which can be constant (i.e., static) or variable (i.e., dynamic). The "state" of an object instance is defined by the values of its attributes at a given point in time. The state of a system is then defined by the set of states of its objects. The operation of the system can then be described in terms of an FSM model, given that "transitions" are defined as changes in the values of object attributes, which changes the state of the system. Transitions occour when events cause the system to perform functions that change the values of object attributes. Exemples of these concepts are described in Section 4 below and the attached figures.

The benefits of the object-oriented FSM modelling approach are as follows:

- it provides a means of completely characterizing a system " in terms of a set of modular objects, and its operation in terms of a finite set of well-defined states and transitions

- it allows the system to be described at an appropriate level of abstraction to model the objects, attributes, functions, and events of interest

- it provides an external (observable) view of a system in terms of the functions it performs in response to events, a view which hides the implementation details of how the system performs these functions

- it can be formally specified

An object-oriented FSM modelling approach is in general appropriate whenever an external view of a system is desired (i.e. , the view of an SSF from an SCF). In particular, it is useful in describing a relationship or an information flow between systems, and is often used to describe protocol machines at a system interface (where protocol messages serve as events). The benefits of this FSM modelling approach serve to meet the general call modelling objectives/criteria described in the IN Baseine Document. In particular, it helps to:

- provide a high-level, vendor/implementation independent abstraction of call/service processing functions

- present an observable view of an SSF to an SCF

- provide a framework for defining the information flows between the SSF and SCF

- provides a framework for ensuring the correct sequencing of functions within an SSF

3 INTERACTIONS

3.1 Introduction

Several types of interactions need to be defined and specified in the SSF to solve the questions of synchronizations and interlocks

- within a subdomain (socket type) between operations on the same object which need to be executed or stopped at specific conditions

- within a domain, between operations and events on the same objects which potentially are controlled by different Socket types. (e.g. a leg in the status monitoring socket)

- within a domain, between instances of similar Functional Entities. In that case the SSF is responsible to properly manage interactions between functions residing in different instances of the FE that are simultaneously active on a single call. Some examples are pictured below.

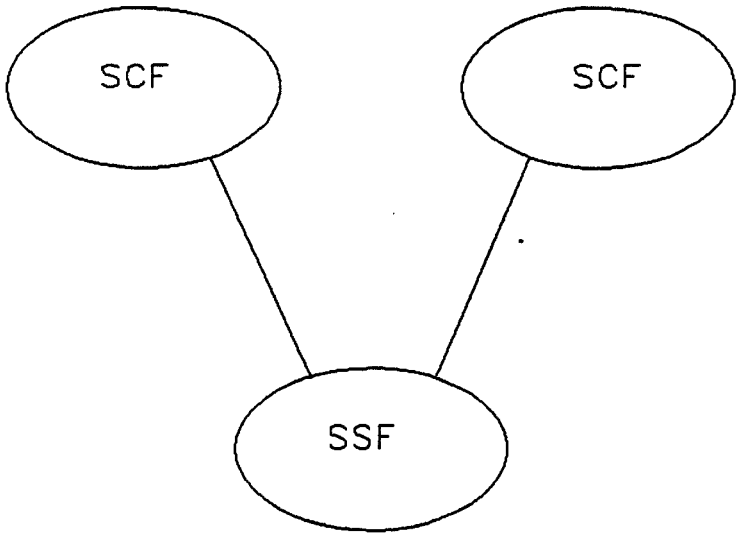


Fig. 3.11. Multiple SCF instances

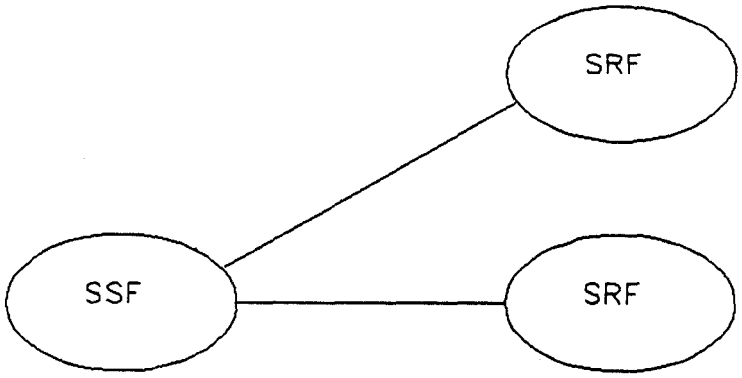


Fig. 3.12. Multiple SRF instances

The reverse case (e.g. One SCF interacting with multiple SSFs) is under the interacting FE responsibility.

3.2 Interactions between operations

The interface between SSF and other FEs is based upon operations, which are requeste from the SSF for some action to be performed by the other FE, or vice versa. This execution must follow some general principles:

1) Each operation received by the SSF has to be processed in a service independent way

2) The order of requested operations has to be respected by the SSF

3) The execution of a given operation doesn't take into account the previously processed operations

The latter statement means that the processing of a given operation doesn't depend directly on the previous operation (even if it implicitly depends on it, because it depends on the states of the switching resources after the processing of the previous operation).

Neverthelesss, this statement only considers the complete processing of operations. What is the matter when an operation is requested by one FE while another one is being processed on the same objects?

Firstly an operation A doesn't interact with an operation B if they don't act upon one or several common logical objects. As soon as A and B attempt to act on the same object, it becomes necessary to define their mode of interaction. We can define five modes of interaction between operations (say B is requested while A is being processed):

1) Parallel execution

   B is started right away upon its invocation, thus A and B are processed in parallel.

2) serial execution

   B is started after the end of A.

3) stop execution

   A is stopped and B is started.

4) no execution

   B cannot be invoked while A is being processed. Then B is rejected.

5) specific execution

   B is started at some context related time.

Each operation may include a parameter "InteractionIndicator" which specifies the interaction mode to be used between this operation and the previous one.

Values of InteractionIndicator:

PA: parallel execution
SE: serial execution
ST: stop execution
NO: no execution
SP: specific execution

The SSF should define, for each couple of (A,B) operations:

1) the default value of InteractionIndicator in B

2) the set of values allowed for InteractionIndicator in B

This can be represented as an interaction matrix.

3.3 Interactions between sockets

F.F.S.

3.4 Interactions between instances of FES

F.F.S.

4 CONNECTION CONTROL DOMAIN (CCD)

4.1 Overview

The model pictured below represent the 5 proceeding entities of the SSF involved in handling interactions with the SCF.

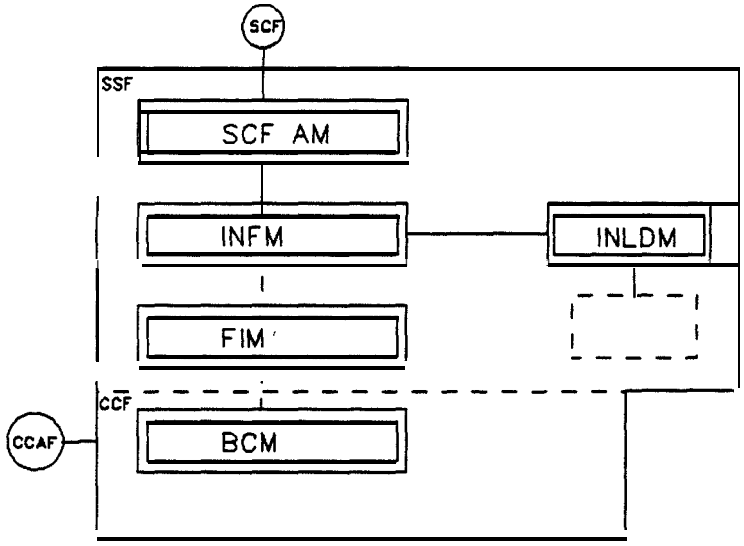


Fig. 4.1. CCD overview

SCFAM: SCF Access Manager
INFM : IN Feature Manager
FIM : Feature Interaction Manager
BCM : Basic Connection Manager
INLDM: IN local Resource Data Manager

A model of the SSF/CCF is shown in Figure 4.1. The purpose of this model is to provide a framework for call modelling subjects with respect to the SSF/CCF.

The aspects of the SSF/CCF model briefly described below include the Basic Connection Manager (BCM), the IN-Feature Manager (IN-FM), the Feature Interactions Manager (FIM), the relationship of the BCM to the IN-FM, and the relationship of the BCM and IN-FM to the FIM. Additional detail is provided in subsequent sections.

- BCM - The entity in the CCF that provides basic connection control to establish communication paths for users and interconnect such communication paths, that detects basic connection control events that can lead to the invocation of IN features or should be reported to active IN features, and that manages CCF resources required to support basic connection control. The BCM interacts with the FIM as described in the FIM description below. The BCN manages one or more instances of a connection.

- IN-FM - The entity in the SSF that interacts with the SCF in the course of providing IN services/features to users. It provides the SCF with an observable view of SSF/CCF feature processing activities, and provides the SCF with access to SSF/CCF capabilities and resources. It also detects IN feature processing events that should be reported to active IN features, and manages SSF resources required to support IN features. The IN-FM interacts with the FIM as described below. The IN-FM manages one or more instances of feature segments.

- FIM - The entity in the SSF that provides mechanisms to manage feature interactions between multiple IN features simultaneously active on a single call and between IN features and non-IN features (e.g., embedded switch-based features) simultaneously active on a single call. The FIM also integrates these feature interactions mechanisms with the BCM and IN-FM to provide the SSF with a unified view of feature processing in the SSF for a single call.

- BCM Relationship to IN-FM through FIM - The relationship that encompasses the interaction between the BCM and the IN-FM. The information flow related to this interaction may not be externally visible and not a subject is required to identify how basic connection processing and IN feature processing may interact.

- BCM and IN-FM Relationships to FIM - The relationships that encompass the interaction between the BCM and the FIM, and the IN-FM and the FIM. The information flows related to these interactions may not be externally visible and not a subject for standardization. However, an understanding of this subject is required in order to unify the BCM, IN-FM and FIM.

Each of the 5 processing entities of the SSF is described in a subsequent chapter, together with a “model” which represents the part of the processing entity which needs to be made visible from/to the outside of the SSF.

The names of those models are slightly different between ETSI and CCITT .

Table 4.1.1. below shows which terms are respectively used in ETSI and CCITT.

In the present issue of this TR are shown only the significant parts where ETSI differs from CCITT as of october 1990.

	Processing entity	Model		Representation
		CCITT	ETSI	
SCFAM	SCF Access Manager	NA	NA	ASEs
INFM	IN Feature Manager	INFSM Finite State	SM Socket Model	
FIM	Feature Interaction Manager	CSM Connection Segment Model	SRM Socket Relationship Model	
BCM	Basic Connection Manager	BCSM Basic Call State Model	BCM Basic Call Model	

overall . . . .
for the model
is CCM Connection
Control Model

fig 4.1.1 CCITT/ETSI naming conventations

Note the acronyms BCM are used twice:
as "Basic Connection Manager"
as "Basic Call Model"
For the rest of this document, BCM is understood as "Basic Call Model"

4.2 SCFAM

4.3 INFM

The "Socket Model" described in part I can be viewed as a window between SSF and SCF through which they can see each other and interact in terms of pre-defined objects and operations on the objects.

The INFM is thus the processing entity of the SSF which has responsibility to manage each of the 4 socket types pretaiming to the Connection Control Domain.

The part of the INFM which is concerned with the traffic management socket type is named TMF (traffic management functionality) and is further described below.

4.3.1 TMF activities

Traffic management mechanisms consist of 2 main activities: decision and execution.

The decision activity is either originated automatically by some process within the network (e.g. the detection of an overload situation in a physical node), or originated by an operator's request: therefore we distinguish automatic traffic control from manual traffic control.

The execution activity is composed of three sub-activities:

- activation, i.e. start the filtering of IN calls, specifying all the filter parameters
- modification, i.e. change the values of one or more filter parameters
- de-activation, i.e. end the filtering.

In an IN-structured network, the decision activity is in principle carried out by the Service control Function (SCF). However, the possibility of having the decision activity performed elsewhere (e.g. in the SSP) is not ruled out, and needs further study.

The activation and modification sub-activities imply an interaction between the SCF and the Service Switching Functions (SSF), while the de-activation sub-activity may or may not imply an interaction (see below). All such interactions can be modelled by the Traffic Management Socket, in terms of objects and operations on the objects. However, it is explicitly stated that such modelling should not represent any constraint on the implementation either of the SSF-SCP interface or on the SSP functionalities.

4.3.2 TMF operation

Note: this is a logical description and not an implementation guideline.

The SSF should keep two Separate lists for IN Traffic Management (TM lists), one for the manual TM filters and one for the automatic TM filters. Each element in a list is an active IN traffic filter and therefore is composed of at least 4 elements: scope, severity, duration, treatment, corresponding to the attributes of the object and to the parameters of the operations (ref. Part I of this T.R). The type attribute or parameter is seen as associated to the list.

The duration element in the lists should contain not only the duration itself but also the time of activation, or, identically, it should contain the time of expiration.

Each time a CREATE operation is issued by the SCF through a socket, the SSF should check the manual or the automatic list (according to the type given in the operation) for an element with the same scope. If such an element exists, either the SSF responds negatively to the operation and does not execute it, or the SSF replaces the values in the list element with the values provided in the operation (in this second case the CREATE would work as a MODIFY too: this is ffs.). If such an element does not exist in the list, a new element is added to the list<l> with the values provided in the operation. The possibility of creation of several filters with the same operation is ffs.

As soon as the expiration time of an active filter is reached, the corresponding element in the TM list is immediately deleted from the list, and possibly an EVENT operation is generated (ffs.).

Whenever the SSF detects any other event, related to traffic management, that has to be notified to the SCF, it issues an appropriate EVENT operation.

Whenever there is at least one active filter in the TM lists, the SSF checks each incoming IN call to see if it has the characteristics specified by any of the active filters, and, if that is the case, whether the call should be blocked or not by the filtering algorithm.

In other words, each call is checked against<2> the TM lists and if a match is found for the scope parameter, the filtering algorithm is applied to that call. Such algorithms are implementation dependent (e.g. percentage blocking, time gapping, etc.).

Therefore, some IN calls that transit through the SSF are blocked by the IN traffic filters, while others are let through. Each call blocked by an IN traffic filter is given the tone or announcement as specified in the treatment element, and possibly increases a counter. However, the blocked calls do not require any real-time interaction between SSF and SCF, as they are entirely treated within the SSF. In the considerations above, the assumption is made that each call only finds one match in the TM lists. In fact it is possible that the characteristics of a call match with the scope values contained in more than one element on the lists. This occurrence (“multiple matches”) has to be investigated.

A solution proposed for multiple matches is the following:

<1> The dimension of the table and all other physical details (e.g. the actual mechanism employed in the switching exchange to filter the calls) are not considered here.

<2> It may be useful to have the first check for TM controls associated with the Trigger Check Point in order to limit the processing load related to such checks: this is however an implementation issue and will not be advised here.

1) (Multiple matches in different lists) - the manual list should be checked first, and as soon as a match is found there, the related filter will be applied but the automatic list will not be checked for that call. This means that manual controls have precedence over automatic controls.

2) (Multiple matches within the same list) - the following simplifying assumptions are made:

   a) both the SAI and SIE parameter formats are digit sequences

   b) either SAI or SEI values can consist of only a partial sequence of digits (i.e. the number of specified digits is not the maximum allowed), but if in one list element the SAI is not fully specified (i.e. it consists of a partial sequence) then the SEI must be void. A new element, that violates this rule with respect to an already existing element, should not be included in the list

   c) with the assumptions above, and considering the concatenation of the SAI and SEI digit sequences as one "code", the longest code has precedence over all the shorter ones that are subset of it. This means that, if within the same TM list there are 2 codes, one n-digits long and the other n-digits long, which are identical for the first n-digits, then the longer code should be checked first and if the call matches with it, the related filter will be applied and the element with the schorter code will be ignored. If the longer code does not match the call parameters, the shorter one will be checked.

4.3.3 Interactions with non-IN traffic management

For the time being, no interactions are specified between IN and non-IN traffic management activities, except for the generation of EVENT operations with appropriate parameters by the SSF.

This point, however, needs further study. Some considerations are contained in Part I of this T.R.

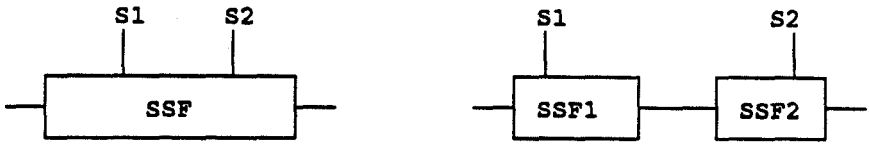
A general requirement on traffic limitation mechanisms, which is applicable to the IN traffic filters, is the following: when one or several filters are active in an SSF, special care should be taken lest particular types of calls, e.g. emergency calls, be blocked by these filters. This should be handled within the SSP and have no influence on the TM socket model, but it is left for further study.

4.4 FIM

Although the need for interaction mechanisms between IN and non-IN features is recognized, the mechanisms proposed in this section only address at this point of time the interactions between IN features.

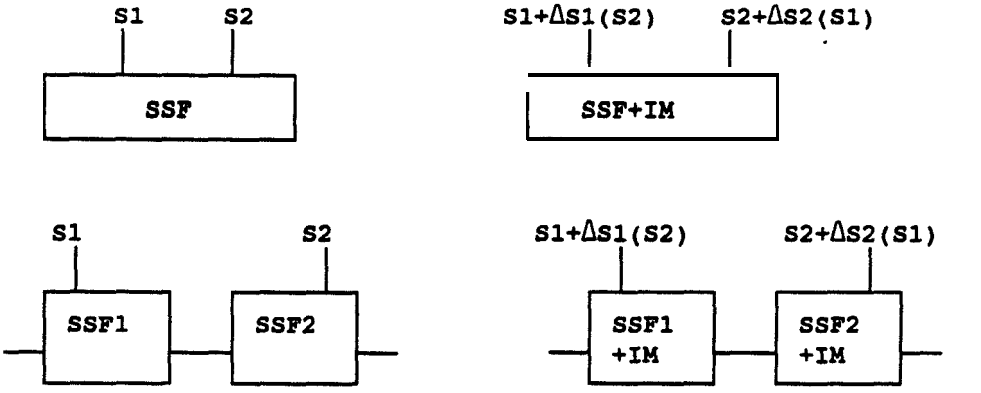
IN and non-IN feature interaction require further study.

The situation we consider is one SSF connected to several service logics acting on a same call (local IN interaction), or several SSFs and several SCFs acting on the same call (distributed IN interaction). How can a SSF manage this situation?



local IN interaction **distributed IN interaction**

Obviously feature interaction needs a certain amount of additional logics, colocated in the SSFs and SCFs. One issue is the location of this additional logic. A reasonable assumption can be stated: **the additional logic for service interaction is located as much as possible in the SSF** (assumption 1). The service independence condition implies that a minimum amount of additional logic is needed in the SCF.



NO INTERCTION SUPPORTED (exclusion S1, S2)

INTERACTION SUPPORTED (IM=interaction mechanism)

Assumption (1) can be summarized : Minimize $\Delta Sl(S2)+ \Delta 2(Sl)$ with IM service independent.

The motivation for such an assumption is that feature interaction is a problem growing very quickly when more and more features are involved, because it is necessary to define for each feature S1: $\Delta Sl(Si,Sj,...,Sk)$ where $l<i,...,k<n$. It becomes rapidly tremendous.

Bearing in mind this assumption, we can go in more details about IM and SCF-SSF interface.

1/Service marks

S1, in order to interact with S2, needs to activate additional logic ΔS1(S2) . Therefore, S1 needs to be reported of the activation of S2. A possible mechanism for doing that is the servicemark (SM). This parameter indicates which feature is activated, and how many times

2/Exclusion table

CC-provide-instruction is sent by the SSF to the SCF when the feature is triggered, and when feature interaction doesn't prevent the feature to be activated.

The SSF handles a table of feature exclusion (exclusion table). For each couple (S1, S2) an exclusion mark indicates whether or not the interaction is allowed. Note this table is not symmetrical (e.g. S1 can be activated while S2 is running, but S2 can't be activated when S1 is running. Moreover, a feature can exclude itself (e.g. a new instance of S1 can't be activated when S1 is running).

Example of an exclusion table:

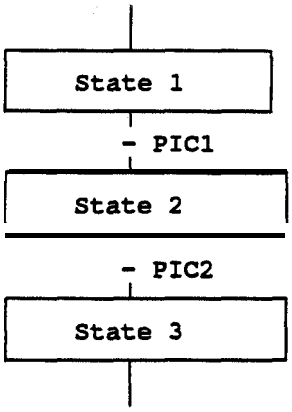
	ΔS1	ΔS2	ΔS3	
ΔS1	N	Y	N	
ΔS2	N	Y	N	
				.....

Thus , when a trigger criteria is fulfilled for triggering S1, the SSF searches for any exclusion with the other features already acting on the same call (Si, i=1...n). If none is detected, CC-provide-instruction is sent to S1 with service marks Si, i=1. . .n.

3/Trigger priority table

A specific case must be considered: case of features to be triggered at the same TCP. A simple mechanism is to order features at one TCP in a priority order (trigger priority table). The first priority feature which is not excluded/filtered is triggered, while the other ones are not.

4/Trigger mark



CC-resume-BCSM allows feature S1, triggered at a TCP, to ask the SSF for resuming the basic call at a resumption point RP. Both TCP and RF are PICs. This RP may be specified by the SCF or the SSF may use a default value, depending on the context.

In case the RP and the TCP are identical (same PIC), other features Si may be triggered at the same TCP. Nevertheless, we can assume that S1 must not be triggered again. To prevent the SSF from doing so, it has to use a trigger mark (TM). For each call instance, a trigger mark indicates which features have been triggered at each TCP.

TCP1 TM=Si,...
TCP2 TM=...

This mechanism allows:

- to trigger more than one feature at a same TCP, despite of the priority mechanism previously described.

- to avoid loops in triggering

Note: the trigger mark and the service mark are not identical, because the trigger mark is local to a TCP. Nevertheless TM -> SM.

4.5 BCM

4.5.1 Introduction

In this section, BCM stands for "Basic Call Model".

The BCM should not be an attempt to describe a standard internal handling of a basic call in an exchange, as that may result in endless discussions, knowing that rather different internal structures exist with the present technology. It should only help to model the external SSF/CCF interfaces i.e. the SSF/CCF - end user (via CCAF), SSF/CCF - SSF/CCF, SSF/CCF - SCF.

For the same reason, only the external view from trigger events, as viewed from outside of the SSF/CCF (by end-users or SCF), needs to be standardized.

It is also important to recognize that some of the transitions, trigger check points and point In Call represented in section 4.5.4. may not be applicable to all types of calls.

This section describes how the functionality contained within the BCM can be described in terms of three "segments". Each segment is represented as an individual finite State Machine corresponding to the terminator and associator segments of the Connection Control Model (CCM). This section also contains a detailed description of each of these Finite State Machines (FSM).

4.5.2 The case for segmentation

The basic notion behind IN is allowing an entity external to the switch (i.e. the SCP) to re-use the basic call processing functionality provided by it. In order for this to be done efficiently and flexibly, the basic call processing functionality must be partitioned into re-usable components. This is so that service writers can easily gain access to those portions of call processing which they wish to use.

If an analysis of the activities involved in setting up a basic call between two parties is made then it can be seen that there are basically three different sets of activities involved which service writers require access to an control over.

a) The activities associated with the detection and authorisation of a request from a user to originate a call.

b) The activities associated with establishing a connection with the destination user. Close control may be required over these activities (e.g. choice of route).

c) The activities associated with presenting a call to the called end-user.

The BCM must be capable of showing the divisions between these different activities.

4.5.3 Representation

As was stated above, the BCM should distinguish between the different sets of activities involved in setting up a basic call between two parties. It therefore allows that the BCSM should be "segmented", and represented as three FSM. Each F is referred to as a "Basic Connection Segment". The Incoming Terminator Segment FSM contains the activities described in 4.5.2a above. The Associator Segment FSM contains the activities described in section 4.5.2b above. The Outgoing Terminator Segment FSM contains the activities described in section 4.5.2c above.

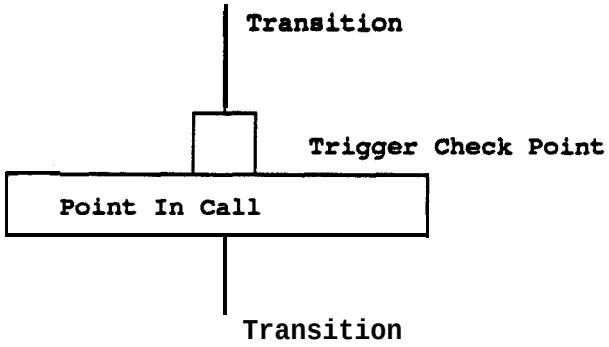
A very important point should be made concerning this division of functionality. The BCM represents an abstraction at a high level of the activities performed by all vendors' switches. The detail of how these things are carried out is far more complicated and up to each vendor. The segmentation of the BCM in to three is ONLY for the purposes of producing a clearer model. It is not meant to imply an implementation.

4.5.4 Segment description

The individual segment Fsms are described in terms of Points in Call (PICs) and Trigger Check Points (TCPs).

PIC represent activities involved in setting up a basic call between two parties. TCPs are placed on the transitions between these activities, and represent points in the setup where IN service logic may become involved in the further processing of the call.

Points in Call and TCPs are represented in Figures II 4-1 to II4-3 as shown below:



The normal path through the FSM will be from top to bottom.

Each PIC is described in terms of:

a) The events which cause entry to this PIC.

b) The activities which are carried out at this PIC.

c) The events which cause an exit from this PIC. If there is a TCP on the transition between this PIC and the next, it is named after that event.

d) Any comments which are applicable to this PIC.

The detailed description of each PIC is given in section 4.5.5.

Where a similar PCI exists in all three segments (e.g. Null) then it is prefixed by the name of the segment in which this states resides (e.g. the Null state in the Incoming terminator is I_Null and the Null state in the Outgoing terminator is O_Null). Also, where a similar event might be received in all three segments the event name is prefixed by either I O or A. (e.g. I_Answer, O_Answer, A_Answer).

Note that the "Incoming" and "Outgoing" terminator segments are defined from the point of view of the associator segment. The “incoming” terminator is, associated with the calling party and the “outgoing” terminator is associated with the called party.

The states shown here have been derived from a consideration of the activities required for a basic POTS call between two parties. The same states would apply to an ISDN basic call, but the “wait time” in some of the states might be zero.

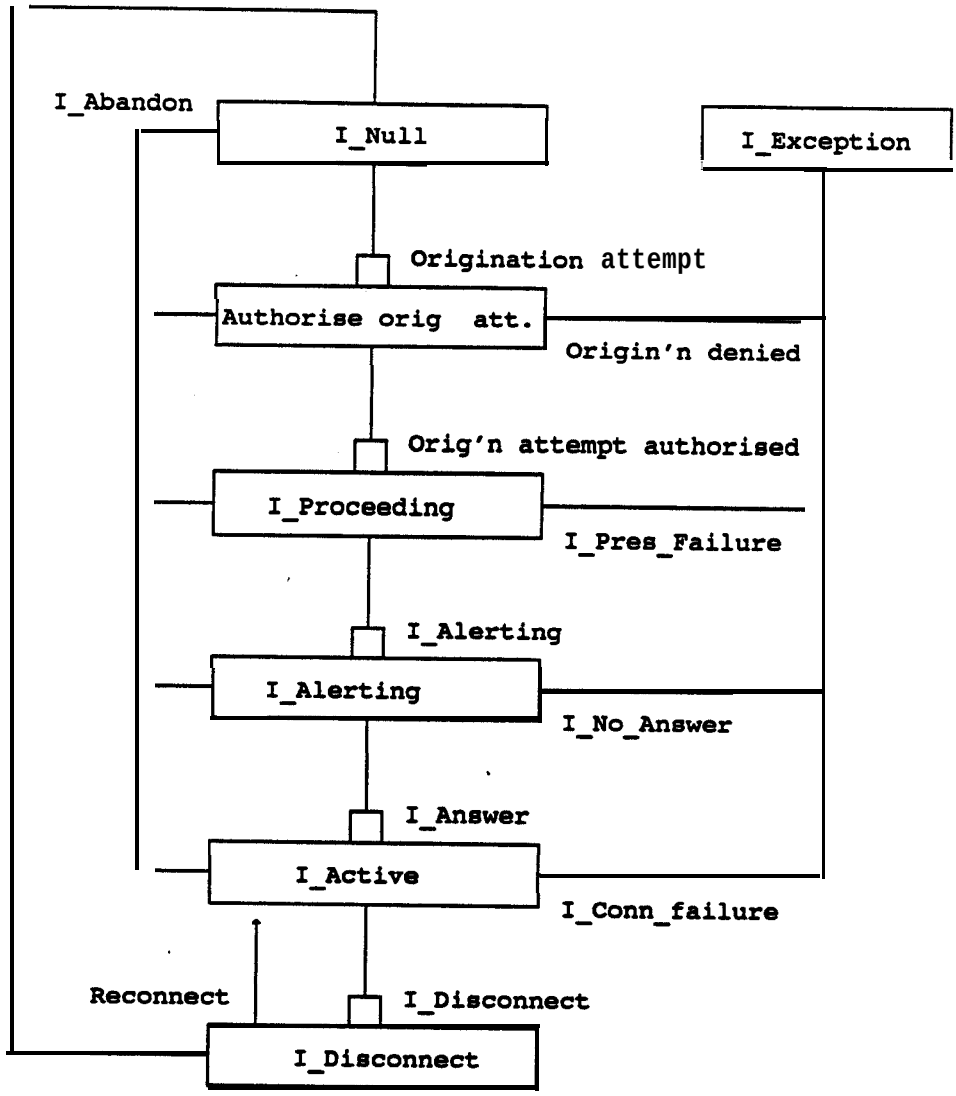


Figure II.4.1 INCOMING TERMINATOR SEGMENT

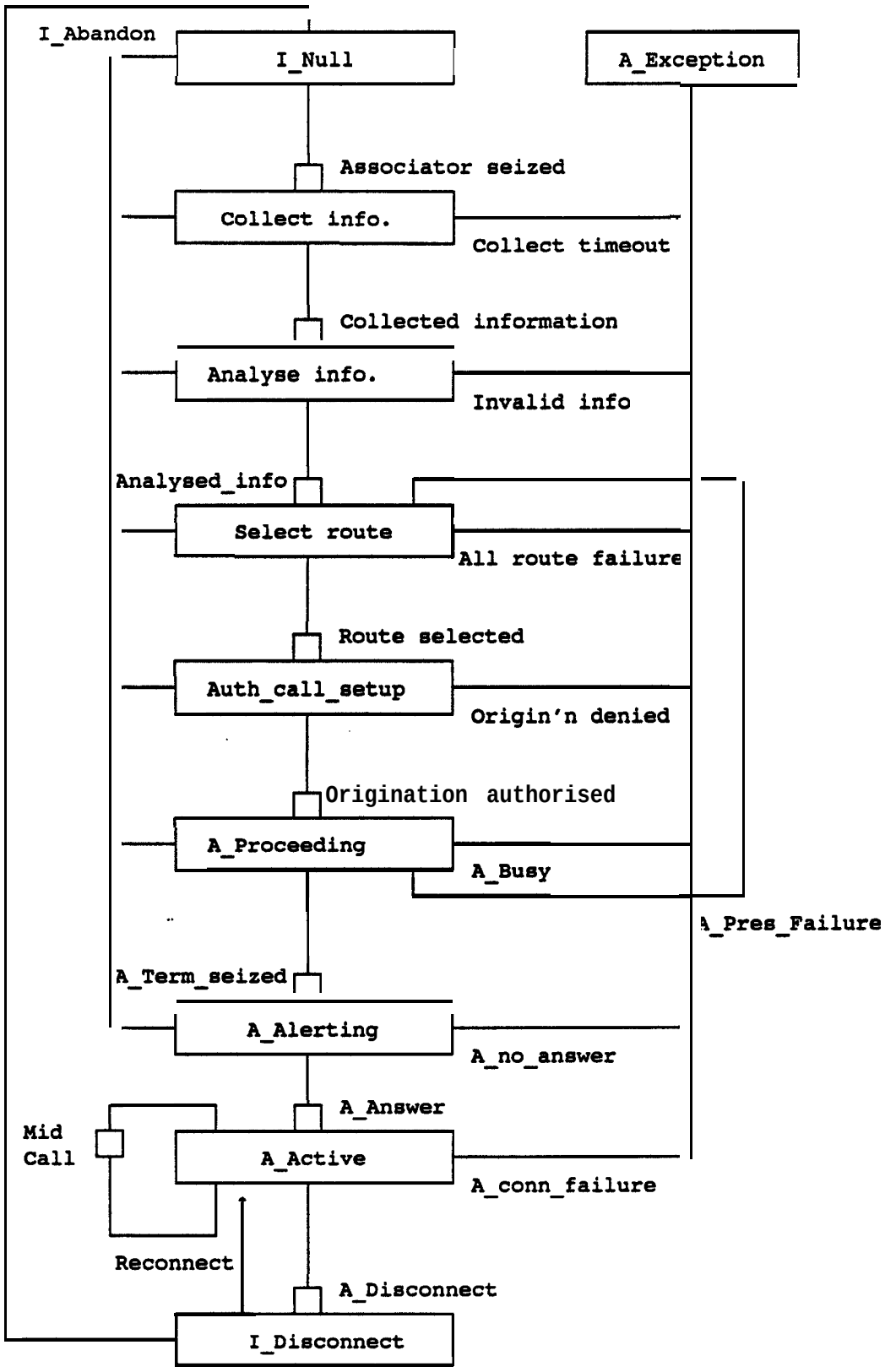


Figure II.4.2 ASSOCIATOR SEGMENT

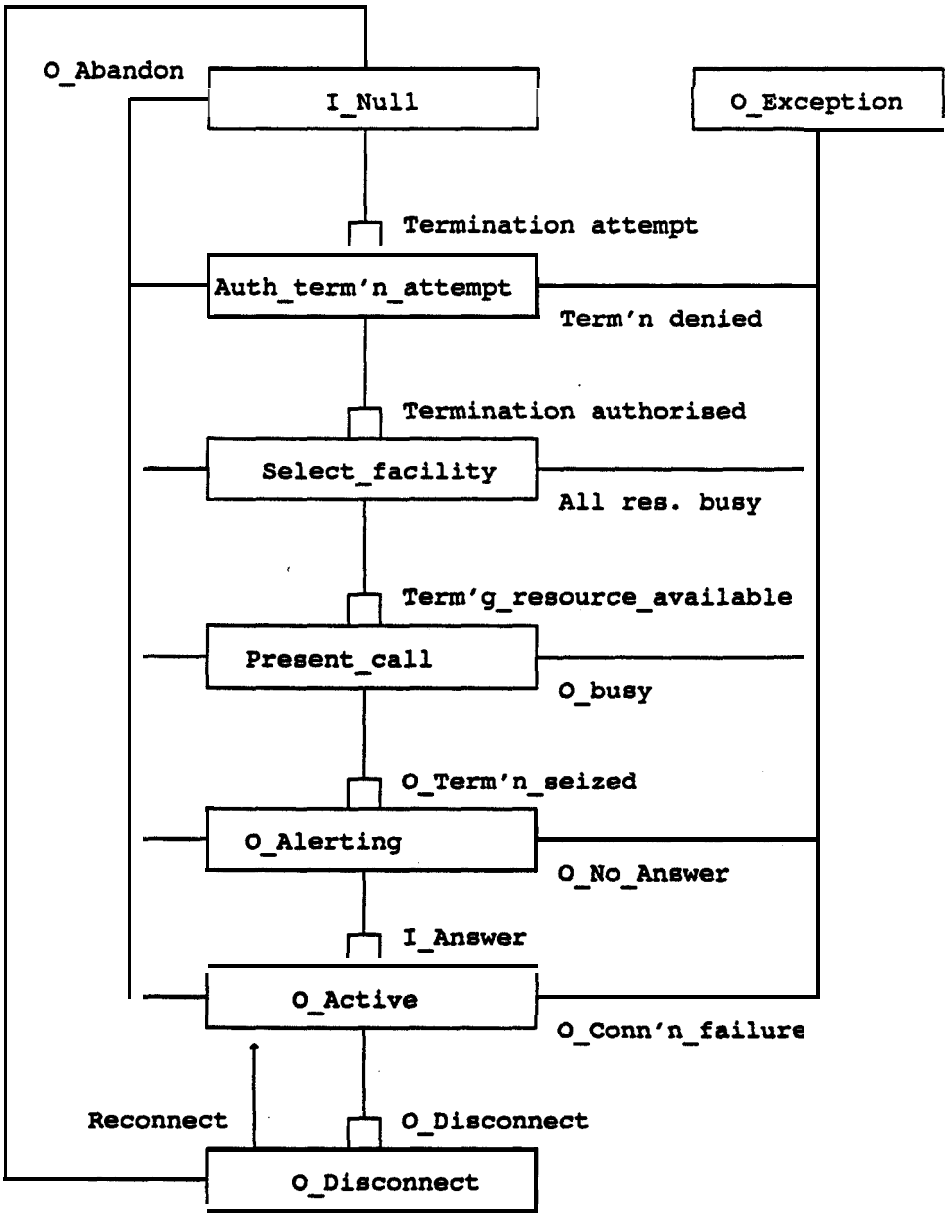


Figure II.4.3 OUTGOING TERMINATOR SEGMENT

4.5.5 Detailed description of PICs

4.5.5.1 Incoming terminator segment

4.5.5.1.1 I_Null

Entry Event: Disconnect and clearing of a previous call.

Action: Interface (line/trunk) is idled (no call exists, no call reference exists, etc.). Supervision is being provided.

Exit Event: Indication of desire to place outgoing call (e.g., off-hook, Q.931 Setup message, ISDN-UP IAM message) TCP - Origination Attempt.

4.5.5.1.2 Autorise_Origination_Attempt

Entry Event: Indication of desire to place outgoing call.

Action: Authority/ability of user to place outgoing call with given properties (e.g., bearer capability, line restrictions) is being verified. The types of authorisation to be performed may vary for different types of originating resources (e.g., for lines vs. trunks).

Exit Events:

- Authority/ability to place outgoing call denied
- Authority/ability to place outgoing call verified (TCP Origination Attempt Authorised)
- Calling party abandons call.

4.5.5.1.3 I_Proceeding

Entry event: Authority to plac outgoing call confirmed.

Action: Continued processing of call setup is taking place by the associator and outgoing terminator segments. The incoming terminator is waiting for an indication from the outgoing terminator segment that the called party is being alerted.

Exit events:

- Notification of alerting has been received via the associator from the outgoing terminator. (TCP I_Alerting)

- Notification is received from the associator that the attempt to present the call to the called party has failed.

- The calling party abandons the call.

4.5.5.1.4 I Alerting

Entry Event: Indication received via the associator that the called party is being alerted of incoming call.

Action: Continued processing of call setup (e.g., power ringing, audible ring indication) is taking place. The incoming terminator is waiting for an indication from the outgoing terminator segment (via the associator) that the call has been answered by the called party.

Exit Events:

- Called party does,not answer within a specified time period.

- Call is accepted and answered by called party (e.g., terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer message received) (TCP I_Answes)

- Calling party abandons calls.

Comment: for inter-exchange calls involving C7, this PIC is entered upon the receipt of an Address Complete (ACM) message.

4.5.5.1.5 I Active

Entry Events:

- Call is accepted and answered by terminating party.

  Action: Connection established between calling an called party. Call supervision is being provided.

  Exit Events:

- A connection failure occurs

- A disconnect indication (e.g., onhook, Q.931 Disconnect message; C7 Release message) is received from either party. (TCP I_Disconnect)

4.5.5.1.6 I Disconnect

Entry Event: A disconnect indication is received form either party.

Action: Disconnect treatment is being applied.

Exit Events:

- Completion of disconnection of call (e.g., expiration of disconnect timing, resources idled)

- Indication that either the called (or calling?) party reconnects before expiration of disconnect timing.

comment: The action taken in this state will be different depending on whether it was the called or the calling party who disconnected first.

4.5.5.1.7 I Exception

Entry Event: An exception condition is encountered

Action: default handling of the exception condition is being provided

Exit Event: For further study.

4.5.5.2 Associator segment

4.5.5.2.1 A Null

Entry Event: Disconnect and clearing of a previous call.

Action: Associator is idle. i.e. it has not been “seized” by a terminator.

Exit Event: Request from a terminator segment (or feature segment) TCP - Associator seized.?

Comment: Whether they should be a TCP on this transition is for further study.

4.5.5.2.2 Collect Information

Entry event: Request from an incoming terminator or feature segment.

Action: Initial information package/dialling string (e.g., service codes, prefixes, dialled address digits) being collected from user. No further action may be required if an en bloc signaling method is in use (e.g. an ISDN user using en bloc signaling, an incoming C7 trunk).

Exit Events:

- Information collection error has occurred (e.g., invalid dial string format, collection timeout)

- Availability of complete initial information/dialling string from user. (This event may have already occurred in the case of en bloc signalling, in which case the waiting duration is this PIC is zero). (TCP Collected Information)

- Originating party abandons call.

Comment: Some digit analysis is required to determine the end of dialling and it may not be possible to separate the two PICS in the manner described here. It is assumed that this analysis may be modelled as separate from the rest of digit analysis. There is no intention to specify an implementation. However, a switch should externally present the separable view described. This PIC may also involve some requests to the incoming terminator segment.

4.5.5.2.3 Analyse Information

Entry Event: Availability of complete initial information/dialling string from user.

Action: Information being interpreted and/or translated according to dialling plan. Routing address and call type (e.g., local, long-distance) are being determined.

Exit Events:

- Unable to interpret and translate dial string in the dialling plan (e.g., invalid dial string)

- Availability of routing address and call type. (TCP Analysed Information)

- Originating party abandons call.

Comment: Note that routing address does not necessarily mean that the final physical route has been determined (e.g., route list has not been searched, hunt groups have not yet been searched, DN has not yet been translated to physical port address), though this may be the case.

4.5.5.2.4 Select Route

Entry Events:

- Availability of routing address and call type.

- Unable to complete call using specified route (e.g. congestion)

hei Action: Routing address and call type being interpreted. The next route is being selected. This may involve sequentially searching a route list, translating a DN into physical port address, etc. The individual destination resource out of a resource group (e.g., a multi-line hunt group, a trunk group) is not selected. In some cases (e.g., a normal POTS line), a single resource (not a group) is selected.

Exit Events:

- Unable to select a route (e.g. unable to determine a correct route, no more routes on route list)

- Terminating resource (group) to which call should be routed has been identified (TCP Route Selected). This corresponds to the identification of a outgoing terminator segment.

- Originating party,abandons calls.

4.5.5.2.5 Authorise Call Setup

Entry Event: Terminating resource (group) to which call should be routed has been identified.

Action: Autority of calling resource to place this particular call being verified (e.g., checking business group restrictions, route restrictions). Authorisation could involve requesting and receiving additional information (e.g. PIN code) from the user. The types of authorisation checks to be performed may depend upon the type of originating resource (e.g., line vs. trunk).

Exit Events:

- Authority of calling party to place this call is denied (e.g., business group restriction mismatch)

- Authority of calling party to place this call verified. (TCP Origination Authorised)

- Originating party abandons calls.

4.5.5.2.6 A Proceeding

Entry Event: Authority of calling party to place call verified

Action: call is being processed by the outgoing terminator segment. The associator segment is awaiting some indication that the call has been presented to the called party.

Exit Events:

- Unable to present call to the terminating party (e. g., network congestion)

- Called party is busy

- Terminating party is being alerted (TCP A_Term Seized)

- Originating party abandons calls.

4.5.5.2.7 A Alerting

Entry Event: Called party is being alerted of incoming call.

Action: Continued processing of call setup (e.g., power ringing, audible ring indication) is taking place. The associator is waiting for an indication from the outgoing terminator segment that the call has been answered by the called party.

Exit Events:

- Terminating party does not answer within a specified time period.

- Call is accepted and answered by terminating party (e.g., terminating party goes offhook, Q.931 connect message received, ISDN-UP Answer message received), (TCP A_Answer)

- Originating party abandons calls.

Comment: This PIC is not applicable when terminating to a non-C7 trunk group. For terminations to C7 trunk groups, this PIC is entered upon the receipt of an Address Complete (ACM) message.

4.5.5.2.8 A Active

Entry Events:

- Call is accepted and answered by terminating party.

- called party reconnects before the expiration of disconnect timing.

Act ion : Connection established between calling and called party. Message accounting/charging data may be being collected. Call supervision is being provided.

Exit Events:

- A connection failure occurs

- A "mid-call" event such as hookflash is received from either party. (TCP Mid_call)

- A disconnect indication (e.g., onhook, Q.931 Disconnect message, SS7 Release message) is received from either party. (TCP A_Disconnect)

4.5.5.2.9 A Disconnect

Entry Event: A disconnect indication is received from either party.

Action: Disconnect treatment is being applied.

Exit Events:

- Completion of disconnection of call (e.g., expiration of disconnect timing, resources idled)

- Indication that either party reconnects before expiration of disconnect timing

4.5.5.2.10 A Exception

Entry Event: An exception condition is encountered

Action: Default handling of the exception condition is being provided

Exit Event: For further study.

4.5.5.3 Outgoing terminator segment

4.5.5.3.1 O Null

Entry Event: Disconnect and clearing of previous call.

Action: Interface (line/trunk) is idled (no call exists, no call reference exists, etc.). Supervision is being provided.

Exit Event:

- Indication of incoming call received from associator. (TCP - Termination Attempt)

- Indication of calling party abandon.

4.5.5.3.2 Authorise Termination Attempt

Entry Event: Indication of incoming call received from associator.

Action: autority to route this call to terminating user is being verified (e.g., business group restrictions, restricted incoming access to line, bearer capability). (It has been suggested that this PIC may not be applicable for terminations to trunks).

Exit Events:

- Authority to route call to specified terminating resources (or group) denied.
- Authority to route call to a specified terminating resource (or group) verified. (TCP Termination Authorised)
- Indication of calling party abandon.

4.5.5.3.3 select Facility

Entry Event: Authority to route call to specified terminating resource group verified.

Action: A particular available resource in the specified resource group is being selected. It is possible that all resources in the group could be busy. A single resource is rated as a group of size 1.

Exit Events:

- All resources in group busy.
- Available terminating resource in resource group identified. (TCP Terminating Resource Available)
- Indication of calling party abandon.

4.5.5.3.4 Present Call

Entry Event: Available terminating resource identified.

Action: Terminating resource informed of incoming call (e.g. line seizure, Q.931 setup message, ISDN_UP IAM message). In the case of an analogue line, power ring is in the process of being applied.

Exit Events:

- Cannot present call (e.g., ISDN user determined busy, ISDN_UP Release message with busy cause)

- Terminating party is being alerted (e.g., power ring being applied, Q.931 Alerting message, ISDN-UP ACM message). In the case of an analog line, this event should occur almost instantly after entering this PIC. (TCO O_Termination_Seized)

- Indication of calling party abandon.

4.5.5.3.5 O Alerting

Entry Event: Terminating party is being alerted of incoming call.

Action: Continued processing of call setup (e.g., power ringing, audible ring indication) is taking place. Supervision is waiting for the call to be answered by terminating party.

Exit Events:

- Terminating party does not answer within a specified duration.

- Call is accepted and answered by terminating party (e.g., terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer message received) (TCP O_Answer)

- Indication of calling party abandon.

Comment: This PIC is not applicable when terminating to a non-C7 trunk group. For terminations to C7 trunk groups, this PIC is entered upon the receipt of an Address Complete (ACM) message.

4.5.5.3.6 O Active

Entry Events:

- Call is accepted and answered by terminating party Action: Connection established between calling and called party. Call supervision is being provided.

  Exit Events:

- A connection failure occurs.

- Disconnect indication (e.g., onhook, Q.931 Disconnect message, SS7 Release message) is received from either party. TCP O_disconnect.

4.5.5.3.7 0 Disconnect

Entry Event: A disconnect indication is received from either party

Action: Disconnect treatment being applied.

Exit Events:

- completion of disconnection of call.

- Reconnection by either party before expiration of disconnect timing.

Comment: Disconnect indications and treatment are asymmetrical in the way disconnect timing is applied.

4.5.5.3.8 O Exception

Entry Event: An exception condition is encountered

Action: default handling of the exception condition is being provided

Exit Event: For further study.

History

Document history	
May 1991	First Edition
November 1996	Converted into Adobe Acrobat Portable Document Format (PDF)