ETSI TECHNICAL REPORT

ETR 023

June 1991

Source: ETSI TC-NA

Reference: DTR/NA-060101

ICS: 33.040

Key words: Telephone networks, intelligent networks

Network aspects
Intelligent Networks: Framework

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

Foreword

This ETSI Technical Report (ETR) has been produced by the Network Aspects (NA) Technical Committee of the European Telecommunications Standards Institute (ETSI).

ETRs are informative documents resulting from ETSI studies, but which are not suitable for adoption as formal standards (ETSs or I-ETSs).

The present document is intended to set out a stable basis for the developments of Intelligent Network (IN) standards.

It contains information regarding IN Concepts and Modelling:

- IN objectives, definition and scope
- IN overall requirements
- IN conceptual model
- Service plane architecture
- Global functional plane architecture
- IN functional architecture (high level)
- IN call modelling (high level)
- OAM (high level)

This ETR records the current status of study into IN within ETSI Sub-Technical Committee NA6 up to September 1990, identifies the aspects which could be standardised, and outlines a workplan (Annex A).

IN will be standardised in phases. The technical capabilities to be standardised in Phase 1 will be identified in a subsequent ETR.

Blank page

1 Objectives, scope and definition of IN

1.1 Motivation, requirements and modelling concepts

1.1.1 Motivation

The term IN (Intelligent Network) is used to describe an architectural concept for all telecommunications networks. IN aims to ease the introduction of supplementary services[1)] (UPT, Freephone...) based on more flexibility and new capabilities.

Currently, the subject of IN is motivated by the interests of telecommunications service providers to rapidly, cost effectively and differentially satisfy their existing and potential market needs for services. Also, these service providers seek to improve the quality and reduce the cost of network/service operations and management.

Additionally, current trends in technology permit and demand greater degrees of intelligence, and freedom in the allocation of this intelligence in the telecommunications network. Factors permitting such intelligence include:

- advances in digital transmission and switching,
- common channel signalling, distributed data processing,
- data base management and expert systems.

Other advances, for example the improved mobility derived from miniaturisation of electronic components, demand greater degrees of distributed functionality within and between service provider networks.

1.1.2 Objectives of IN

The objective of IN is to allow the inclusion of additional capabilities to facilitate service/network implementation independent provision of services in a multi-vendor environment. Service implementation independence allows service providers to define their own services independently of service specific development by equipment vendors. Network implementation independence allows network operators to allocate functionality and resources within their networks and to efficiently manage their networks independently of network implementation specific developments by equipment vendors.

1.1.3 Scope of IN

Types of networks: IN is applicable to a wide variety of networks including: public switched telephone network (PSTN), mobile, packet switched public data network (PSPDN) and ISDN.The types of networks and services that will be supported will be identified in a subsequent ETSI Technical Report.

A number of factors that may influence the scope of the work on IN were identified as:

- activities on ONP;
- network operator plans;
- industrial product offerings;
- market needs.

---

1) The term supplementary service is applicable to both ISDNs and non-ISDNs (see Glossary)

1.2 Definition of IN

Intelligent Network (IN) is an architectural concept for the creation and provision of telecommunications services which is characterised by:

- Extensive use of information processing techniques;
- Efficient use of network resources;
- Modularisation of network functions;
- Flexible allocation of network functions among physical entities;
- Portability of network functions among physical entities;
- Standardised communication between network functions via service independent interfaces;
- Access to the process of composition of services through the combination of network function;
- Customer control of some of his specific service attributes;
- Standardised management of service logic.

1.3 Evolution of IN

The specification and deployment of networks that meet all of the objectives and comply fully with the definition of the target IN will take many years and will be the subject of a long term architecture study. A minimum set of criteria for identifying AN INITIAL STANDARDISATION PHASE could be identified. For instance:

- a distributed intelligence with unified network control,
- with service logic and parameters controllable by the service provider, and
- specified with sufficient open-ended features so as to gracefully evolve or expand. One important open-ended feature may be the independence between service control capabilities and bearer networks, e.g. N-ISDN and B-ISDN.

Identification of the complete operative criteria set is a subject for further study.

Even with such "minimum criteria", it is recognised that to effectively plan evolution beyond the INITIAL PHASE, longer-term target architecture(s) must be considered. However, with numerous technical and market uncertainties, longer-term target architecture(s) are themselves likely to evolve.

Of course, the smooth transition from current architectures to the INITIAL PHASE should be considered when specifying the INITIAL PHASE using existing Recommendations whenever possible.

The evolution needs and attributes of IN, as listed below, could constitute part of the complete operative criteria set:

[1] An essential attribute for IN evolution is the modularisation of functionality and information, thereby providing flexibility in its distribution while maintaining unified control giving a single integrated view. This will facilitate the introduction of new technology as transparently as possible, thus enhancing network performance consistent with quality of service and network integrity objectives;

[2] IN should facilitate cost-effective evolution from existing network bases in practical flexible phases;

[3] In the initial phase(s), network elements may exist that support a limited range of IN functions. This will enable networks to retain existing network equipment initially, while still taking advantage of IN capabilities;

[4] IN techniques will evolve and will be applicable to appropriate types of network. In its continuing evolution, IN will be capable of providing services with mobility functions, ISDN and B-ISDN based services, and circuit, packet and hybrid type services in monomedia and multimedia environments;

[5] As IN evolves in the long term, it will involve non-traditional network elements but may also apply to the traditional ones;

[6] IN evolution will facilitate the integration of a wide variety of services and networks, e.g. circuit/packet/channel-switched services, signalling/transport/ OA & M networks;

[7] The existing wide variety of interfaces (users-users, users-networks, networks-networks, etc.) will evolve and will be rationalised. A single, OSI-aligned protocol architecture may then be used on standard, service-independent interfaces as IN continues to evolve in the long term;

[8] IN will evolve from the initial set of support tools and facilities needed for the creation, introduction and management of services;

[9] IN must inherit and enhance the architectural concept of existing ETSI/CCITT standards;

[10] The co-existence of, and interworking between new techniques and existing techniques for providing services is essential.

The practical evolution towards the Intelligent Network requires two closely related activities. The first addresses what can be done within constraints imposed by existing network capabilities and software structures, and what can reasonably be accomplished in the near term. This work will enable maximum use of existing investment. This work is also expected to prove very useful in defining precisely what it is that cannot be readily accomplished within the constraints of existing network structures and capabilities.

The second, largely parallel, activity identifies new technologies, in particular software technologies, that can meet needs identified but not attainable within presently deployed network structures. The examination of new technologies will be undertaken with a much better overall view in that the problems to be solved are based on the first activity. This aspect of the work may be expected to continue for some significant time as technology evolves, for, at each point where a set of specifications is prepared, that becomes the starting point for the next evolutionary step.

1.4 Relationship with CCITT studies

Working parties within CCITT SGXI and SGXVIII are working on the subject of IN.

In order to influence the content of the evolving CCITT Recommendations the following strategy has been adopted:

- NA6 should try to lead CCITT studies on IN and should contribute to CCITT with a view of maximising the impact of ETSI work in CCITT;

- Stable technical decisions by CCITT should however be taken into account by NA6.

1.5 Relationship with other ETSI studies

NA6 identified relationship with the following ETSI Sub Technical Committees:

NA1: Service descriptions and service description methodology for the IN;

NA2: Numbering, addressing and routing in the IN;

NA4: IN architecture/reference configurations, OA&M/TMN aspects, service and network management;

NA5: IN concepts in a Broadband environment;

NA7: Universal Personal Telecommunication;

SPS1: Network-interconnection and signalling;

SPS2: Signalling network and mobility applications;

SPS3: Requirements for/specification of switching system functionalities;

SPS5: Customer access to the ISDN;

GSM: Digital Mobile Cellular Radio supported by IN architecture;

RES: UMTS (ad hoc group);

RES3: DECT;

RES5: TFTS.

Since NA6 is responsible for coordination of work on IN, these groups are requested to liaise with NA6 on IN matters and seek to ensure that their work is consistent with the overall framework studies of NA6.

1.6 Workplan

The IN workplan is specified in Annex A (informative), of this document.

2 IN functional requirements

2.1 Introduction

IN functional requirements arise as a result of the need to provide network capabilities for both:

[1] the customer needs (services) and

[2] the network operator needs.

A service user is an entity external to the network that uses its services. A service is that which is offered by an Administration or RPOA to its customers in order to satisfy a telecommunications requirement. Part of the service used by customers may be provided/managed by other customers of the network.

Categories 1 and 2 are referred to as "service requirements" and "network requirements" respectively. Service requirements will assist in identifying specific services that are offered to the customer. These service capabilities are also referred to as (telecommunication) services. Network requirements span the ability to create, deploy, operate and maintain network capabilities to provide services. The categorisation of service requirements versus network requirements is schematically shown in figure 1.

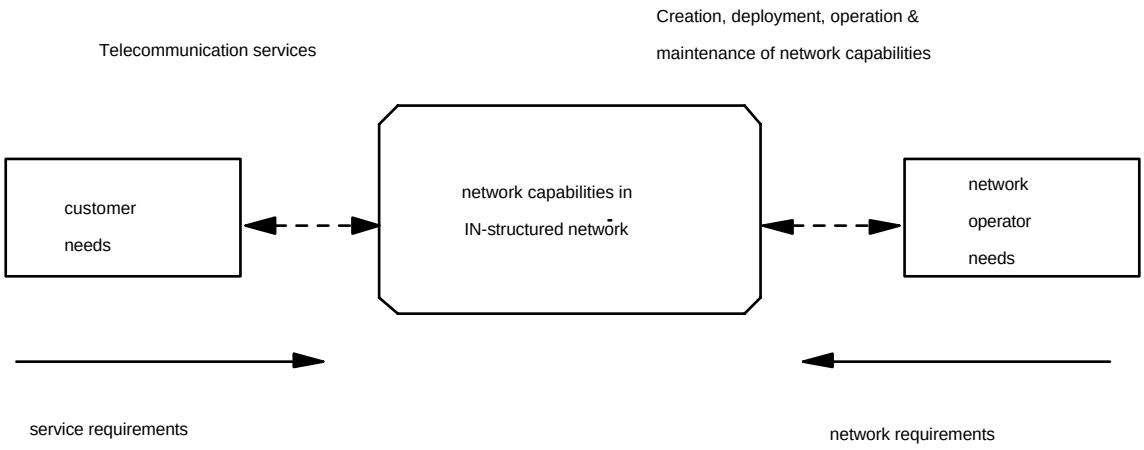


Figure 1: Service requirements vs. network requirements

Service and network requirements can be identified for the following areas of service/network capabilities: service creation, service management, network management, service processing and network interworking.

Service creation

Is an activity whereby supplementary services are brought into being through specification phase, development phase and verification phase.

Service management

Is an activity to support the proper operation of a service and the administration of information relating to the user/customer and/or the network operator. Service management can support the following processes: service deployment, service provisioning, service control, billing and service monitoring.

Network management

Is an activity to support the proper operation of an IN-structured network.

Service processing

Consists of basic call and supplementary service processing which are the serial and/or parallel executions of network functions in a co-ordinated way, such that basic and supplementary services are provided to the customers.

Network interworking

Is a process in which several networks (IN-IN or IN-non IN) cooperate to provide a service.

Figure 2 gives a general overview of these capability areas including their relation to service and network requirements. The network interworking capabilities are not shown in this figure as these are indirectly contained in the other capability areas.

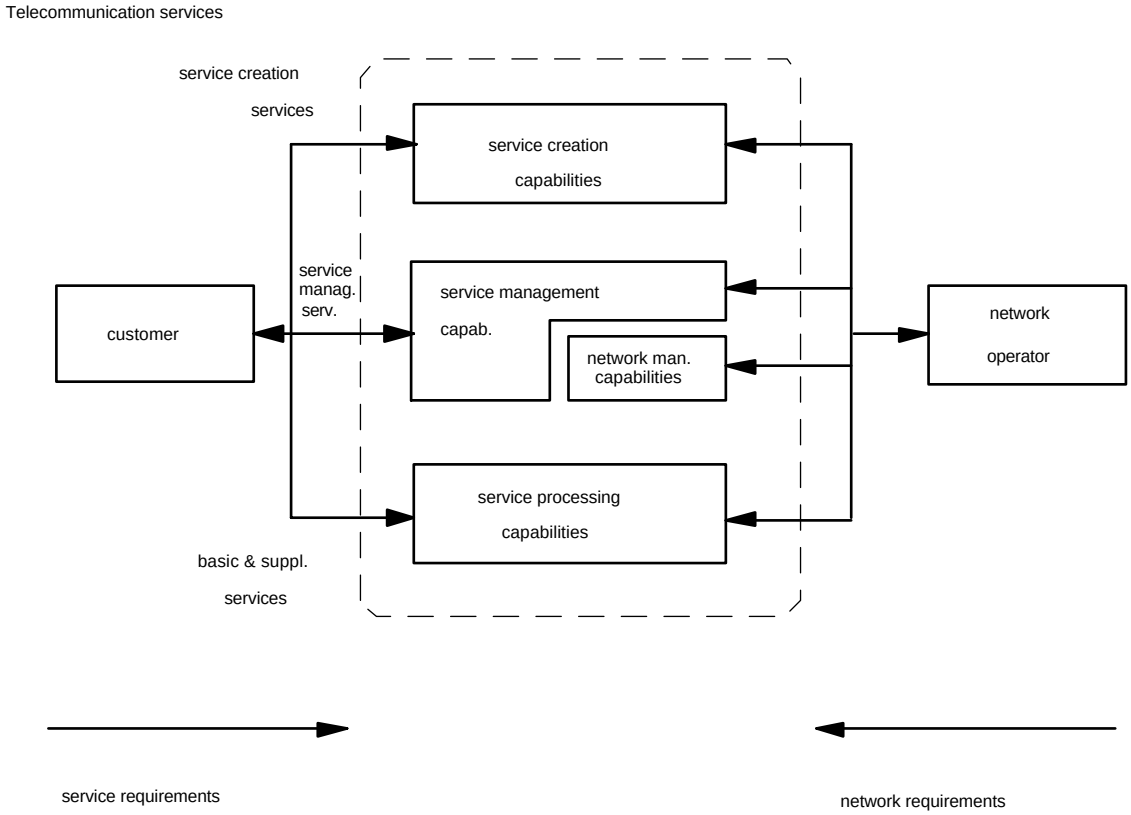


Figure 2: Network capabilities IN in-structured network

2.2 Service requirements

2.2.1 Overall requirements

The following requirements may also apply to existing networks. Nevertheless, they are stated here to underline their importance when defining the IN architecture:

- it should be possible to access services by the usual user network interface;
- it should be possible to access services that span multiple networks;
- it should be possible to invoke a service on a call by call basis, or for a period of time, in the latter case, the service may be deactivated at the end of the period;
- it should be possible to perform some access control to a service;
- it should be easy to define and introduce services;
- it should be possible to support 2 party or multi party services, in the network and in the service creation environment;
- it should be possible to monitor service usage in the network (service supervision, tests, performance information, charging, tariff);
- it should be possible to guarantee the integrity of the network when a new service is being introduced;
- it should be possible to provide services that imply the use of functions in several networks;
- it should be possible to control the interactions between different invocations of the same service.

2.2.2 Service creation

Service requirements for service creation refer to the network capabilities that are necessary for the provision, from a customer point of view, of service creation services to customers. This is schematically shown in figure 3.

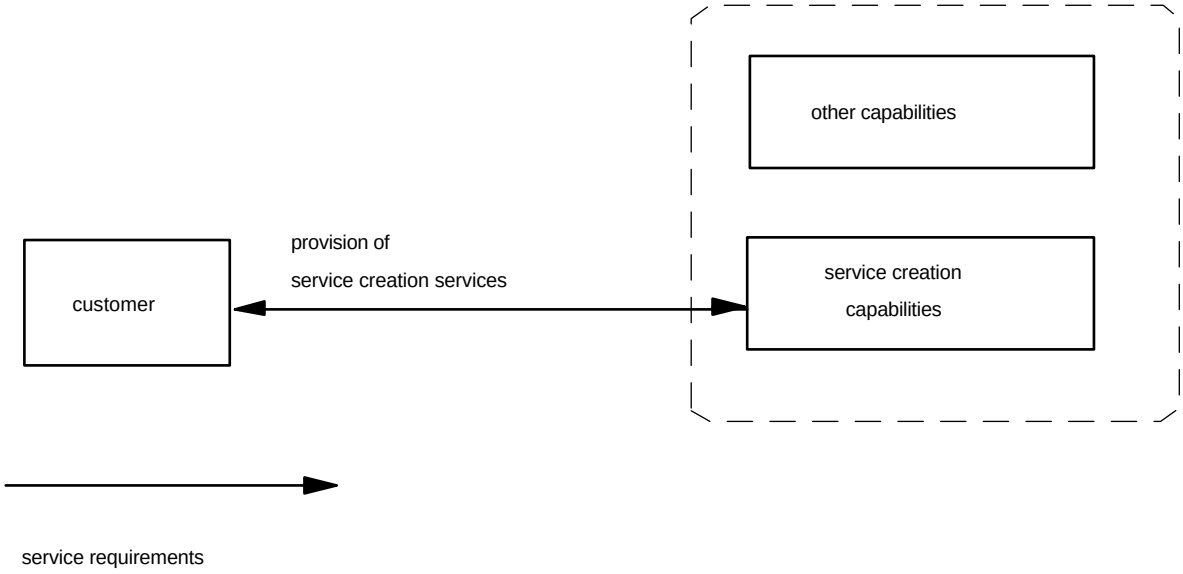


Figure 3: Service requirements for service creation

A subset of the service creation capabilities used by the network operator (described in subclause 2.3.2) may be offered to customers. This is for further study.

2.2.3 Service management

Service requirements for service management refer to the network capabilities that are necessary for the provision, from a customer point of view, of service management services to customers. This is schematically shown in figure 4.

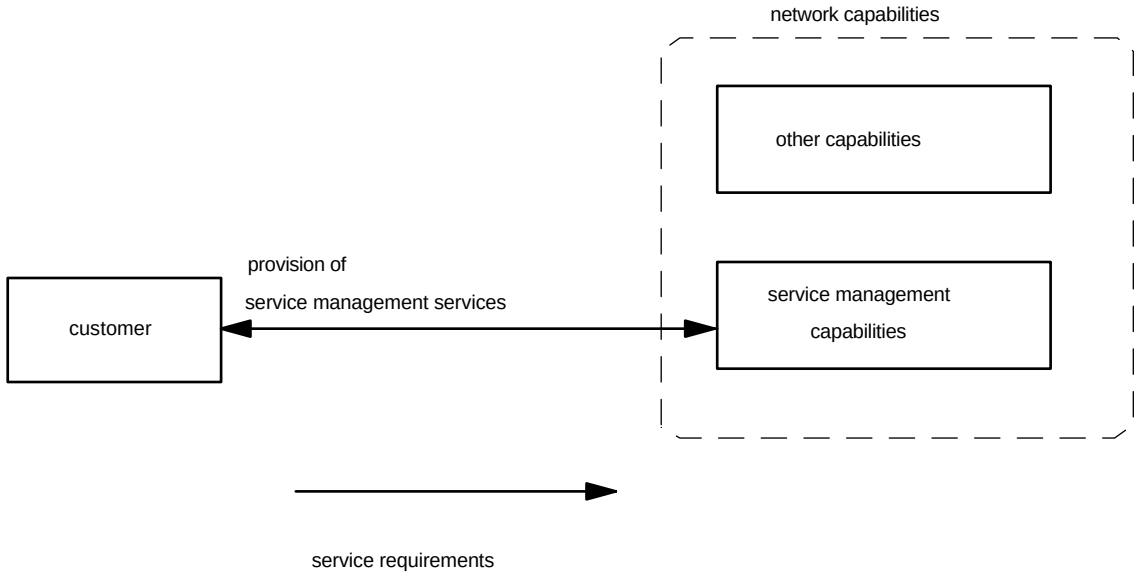


Figure 4: Service requirements for service management

A subset of the service management capabilities used by network operator may be offered to customers.

2.2.3.1 Service deployment

(For further study)

2.2.3.2 Service provisioning management

Service provisioning is the activity of installing and deploying the necessary software in appropriate network elements to realise the functionality of a service to a specific customer along with the initial activation and customisation. After provisioning the customer is administrated in the network as a user of the service.

2.2.3.3 Service control management

This activity includes activation, deactivation, service maintenance and service customisation, after the service provisioning has taken place. Service activation is the activity to make the service usable by a specific customer (e.g. call forwarding activation). Service customisation is the activity of setting up the appropriate service parameters to control the operation of service to meet the specific needs of the customer (e.g. setting the percentage of ACD service).

2.2.3.4 Billing

This activity is mainly to collect data on service usage ant to generate reports thereon for billing either on demand or automatically. It includes the alteration of charge within the framework of agreement with a network operator. Other requirements are the capabilities to customise the formatting and delivery of billing related reports and to receive billing data for customer use.

2.2.3.5 Service monitoring

This activity provides the capability to collect and accumulate statistics on a given service with a view to determine the quality of service operation and adjust the operation to suit the prevailing conditions; also the data may be used in the process of service creation to match ones performance requirements.

2.2.4 Service processing

Service requirements for service processing refer to the network capabilities that are necessary for the provision, from a customer point of view, of basic and supplementary services by an IN-structured network. This is schematically shown in figure 5.

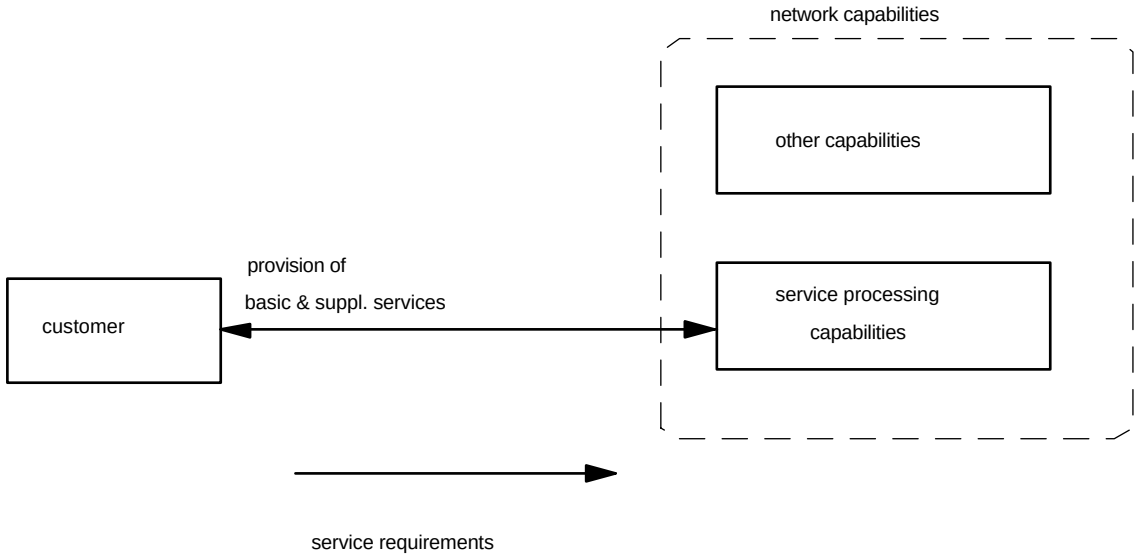


Figure 5: Service requirements for service processing

The IN is primarily a network concept that aims for efficient creation, deployment and management of supplementary services that enhance basic service. With regard to the provision of basic and supplementary services itself the IN concept is "transparent" to the customer, i.e. the customer is unaware of whether a service is provided in an IN-way or not. This "transparency" basically implies that, from a customer point of view, no service processing requirements can be identified that have specific reference to the IN as such. Notwithstanding this, the IN should be capable of supporting a broad range of basic and supplementary services. Service processing requirements can be identified for the following capabilities:

- service capabilities that are necessary to support a broad range of basic and supplementary services;

- access capabilities that are necessary to interface with the network and to access the services;

The relationship between service capabilities and access capabilities is visualised in figure 6.

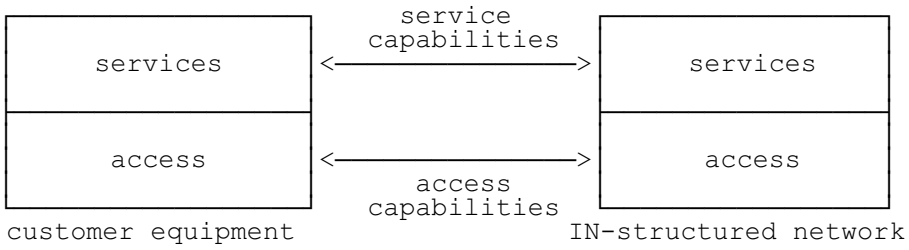


Figure 6: Service capabilities and access capabilities

Service capabilities:

Requirements on service capabilities directly result from the services that are to be supported by the network. Service capabilities are related to basic services (bearer and tele-) and, in particular, to the supplementary services that may enhance these basic services. As stated above, the IN is primarily a network concept for the support (creation, deployment, etc.) of supplementary services. Because supplementary services can only be provided in combination with basic services, it is necessary to define for which basic services the IN concept can be applied. In this respect the IN concept can be applied to the support of supplementary services for the following basic services:

Bearer services:

- circuit-mode unrestricted (various bit rates)

- circuit-mode speech

- circuit-mode audio

- packet switched data services

- circuit switched data services

- B-ISDN services (conversational, messaging retrieval & distribution)

- others.

Teleservices:

- telephony

- telefax

- videotex
- others.

With regard to the provision of supplementary services the network should support service-independent network functions, such that various supplementary services can be constructed from a limited set of network functions (= one of the main IN-objectives). A major task is to properly identify and define these service independent network functions. As each of the above-listed basic service may have its own "range" of supplementary services, each basic service may require its own set of service-independent network functions. Therefore a methodology is required to identify, for each basic service individually, the set of service independent network functions. Possibly, representative supplementary services (existing & future) can serve as a starting point for such methodology.

Access capabilities:

In order to use a particular service, the customer needs an access arrangement to the network(s) that is providing the service. Nevertheless, the customer is mainly interested in the services themselves and not in the specific access arrangement that is used to physically connect to the network. For example, it should be possible that a single Virtual Private Network service can be accessed by various access arrangements such as ISDN interfaces, POTS interfaces, "mobile" interfaces, etc. In this case the VPN service could span multiple networks with different interface technologies. Access arrangements via (non-IN) subnetworks should also be supported (see figure 7). In case of multiple access arrangements to a service, it should be noted that a particular access arrangement may impose technical, operational and/or regulatory limitations on the provided service.

In order to allow freedom of access to services, it is necessary that there is a sufficient degree of independence between the service capabilities on the one hand and the access capabilities on the other hand. The following access capabilities are foreseen for the IN:

fixed network:

- PSTN access
- ISDN access
- PSPDN access

private network access

mobile network:

- PLMN access

broadband network:

- ATM access
- STN access

When a particular service spans multiple networks, network interworking arrangements will be required between the different networks to allow ubiquitous provision of the service. Requirements for network interworking capabilities are identified in subclause 2.3.6.

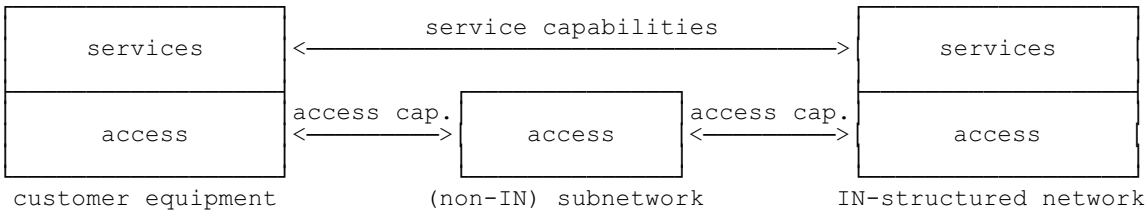


Figure 7: Access to services via (non-IN) subnetwork

2.3 Network requirements

2.3.1 Overall requirements

The following reflects overall requirements. Specific detailed requirements can be found in the relevant subclause.

- it should be possible to evolve cost-effectively from existing network bases to target network bases in practical and flexible manner;
- it should be possible to reduce redundancies among network functions to physical entities;
- it should be possible to allow for the flexible allocation of network functions to physical entities;
- there is a need for communication protocols that allow flexibility in the allocation of functions;
- it should be possible to create new services from network functions in a cost and time efficient manner;
- it should be possible to manage network elements and network resources such that quality of service and network performance can be guaranteed.

2.3.2 Service creation

Network requirements for service creation refer to the network capabilities that are necessary, from a network operator point of view for the creation of new supplementary services. This is schematically shown in figure 8.

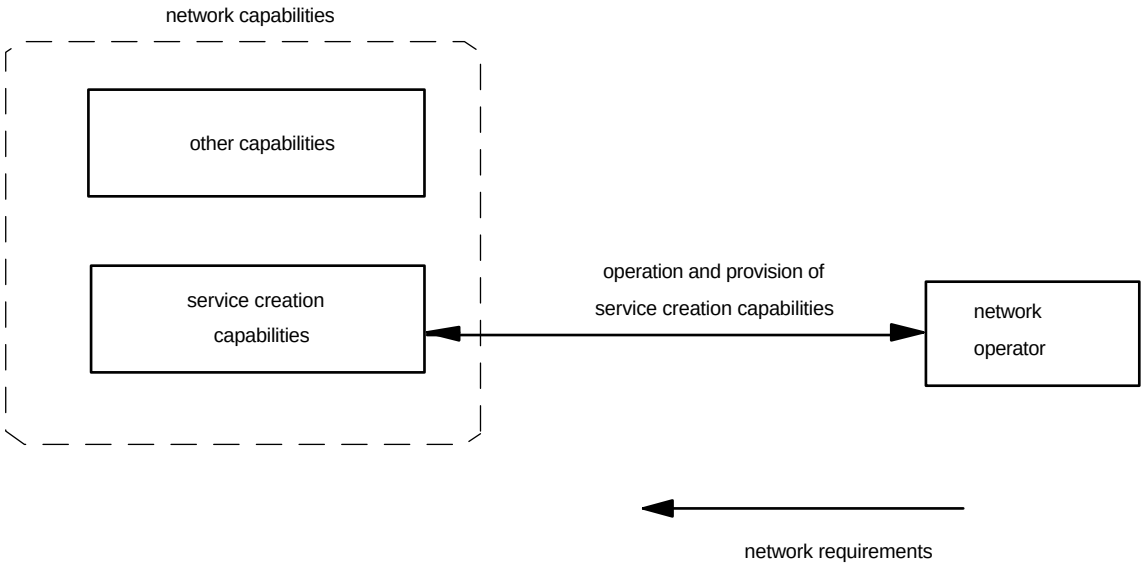


Figure 8: Network requirements for service creation

The creation of (new) services consists of several steps which are summarised in the service creation process. The different steps are:

- service specification;
- service development;
- service verification.

Service specification

Is the first step in the service creation process. As such, this step includes such activities as refinement of detailed service description requirements, functional analysis, generation and verification of a service specification and definition of a high level structured design. The primary outputs of this step include a service specification and a high level structured design which together provide sufficient detail to drive subsequent service development and verification steps.

Service development

Is the step which transforms a high level structured design into a detailed structured software design and subsequently develops the necessary software components, data definitions, etc. required to realise that design. The major output of this step is the developed service software and documentation which is ready for more rigorous service verification testing.

Service verification

Is the step in the Service Creation process where the developed service software (including supporting documentation) is rigorously tested to validate that the resulting service application completely satisfies the requirements specified in the service specification. The principal output of this step is thus the verified service software and supporting documentations required for deployment.

The SCE should, in an efficient and effective way, provide service-implementation independent tools, techniques, languages (e.g. specification languages) and procedures to support the service creation process in which service scripts, represented for example by Service Logic Programmes, Service Management Programmes and Data Template can be created. There is a need to standardise a representation of service logic with standardised functions calls (Application Programming Interface) to IN capabilities.

Thus the SCE provides an environment which allows the easy creation of (new) services in a network-configuration and network type independent way by means of service independent building blocks.

In order to guarantee the integrity and security of the networks as well as to guarantee the integrity of each created service, it should be necessary to define the scope of accessible IN capabilities through the service creation activity. This scope isolates the unnecessary and unsolicited interactions between service and accesses to the network capability.

Each service can be represented by several different types of service scripts. Each script may be classified according to certain characteristics, based on its type of usage. These scripts must all be implemented on a service-, network configuration-, and network type (e.g. PSTN, ISDN, PLMN) independent way. This can best be done by means of service independent building blocks (e.g. function calls via an Application Programming Interface). These building blocks are glued together by means of sequential and conditional programming statements using any programming language (for further study).

Since different scripts can be identified also different building blocks (related to the scripts) can be identified.

Examples of service scripts are:

Service Logic Program (SLP): an SLP consists of for example service execution logic, statistics logic, database logic, charging logic, etc. SLP is a "real-time" program. An SLP logs information concerning billing and statistics into a real-time database.

Service Management Program (SMP): An SMP consists of for example statistics logic, charging logic, database logic, etc. SMP is a "batch oriented" program which, for example, create enhanced charging and service statistics reports derived from the logs created by an SLP. SMP only supports the Service Management and not the Network Element or Network Management.

Data Template Program (DTP): a DTP consists of service-data related logic, user-data related logic, database related logic, etc a DTP is an "interactive" program, which interacts with the network operator (creating and using service-data templates) or the customer (creating and using user-data templates) and updates the service and user data.

Network operators may choose to offer a subset of the service creation capabilities to service users. If this is the case the following requirements apply:

- it should be able to define user classes (e.g. some user can create and use network wide services, some users can create and use local services closed in their own local area, some users can create and use ATM related services, and some users cannot, etc.).

- in order to define service user classes, the restriction is defined with a particular set of network resources (restricted network resources) for particular users. Network resources have different levels of abstraction. The definition of these abstraction needs further study.

2.3.3 Service management

Network requirements for service management refer to the network capabilities that are necessary, from a network operator point of view, to support the proper operation of services. This is schematically shown in figure 9.

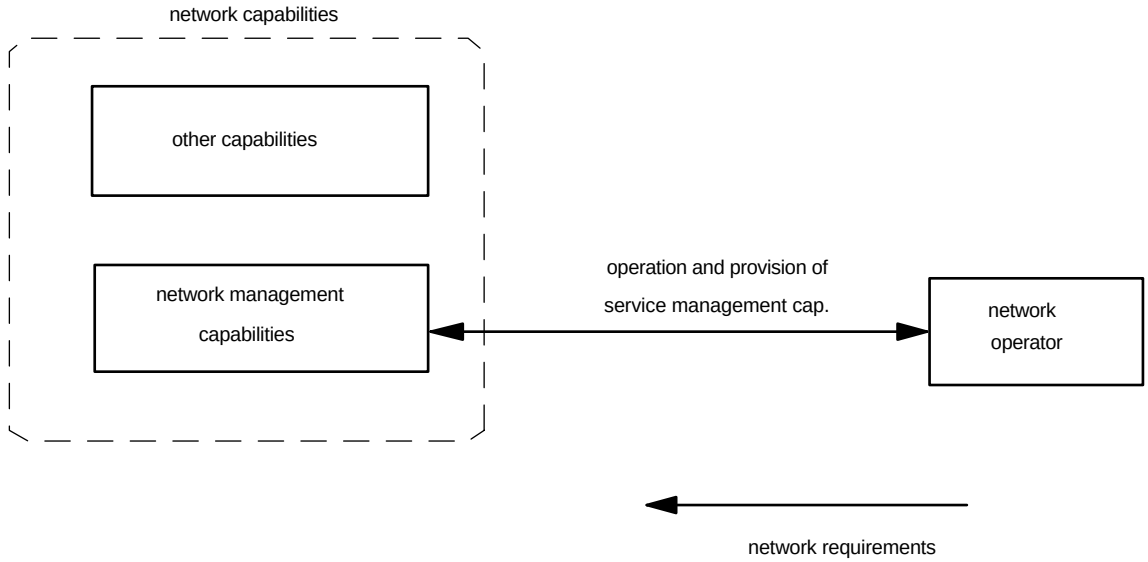


Figure 9: Network requirements for service management

2.3.3.1 Service deployment

This activity requires the following:

- update of network elements with new service procedures; e.g. service logic, operations logic, etc
- deployment of service for limited testing,
- capability of downloading software, of initiating file transfer, of receiving error reports on the said activities and of activating or deactivating a piece of software.

2.3.3.2 Service provisioning

This activity involves the administration of a customer to a new service. This may require the initialisation of trigger table and the creation of customer's service profile in the service logic.

2.3.3.3 Service control

Customisation of data for service operation is validated in this activity. Access control is exercised by the examination of access rights; e.g. customer access of network data for customisation.

This activity also performs general parameter checking when customers access data; e.g. non-existent numbers. Also customer or service profile verification is performed. Service control may result in the modification of trigger table entries and/or service profile of the customer.

2.3.3.4 Billing

This activity performs service independent billing operation such as modification of tariffing, charging, bill determination and generation and storing of billing records. Different types of charging are to be considered; e.g. flat rate, usage sensitive, etc. Network resources used for providing a service can also be charged under this activity.

2.3.3.5 Service monitoring

The following are considered under this activity:

- capability to monitor, delete, define the frequency of schedule, process of and report on measurements statistics);
- collect and validate performance data, status information and network configuration either on demand or automatically;
- storage of collected data and use for analysis if required;
- initiate observations or measurements using service management.

2.3.4 Network management

Network requirements for network management refer to the network capabilities that are necessary, from a network operator point of view, to support the proper operation of the IN-structured network. This is schematically shown in figure 10.

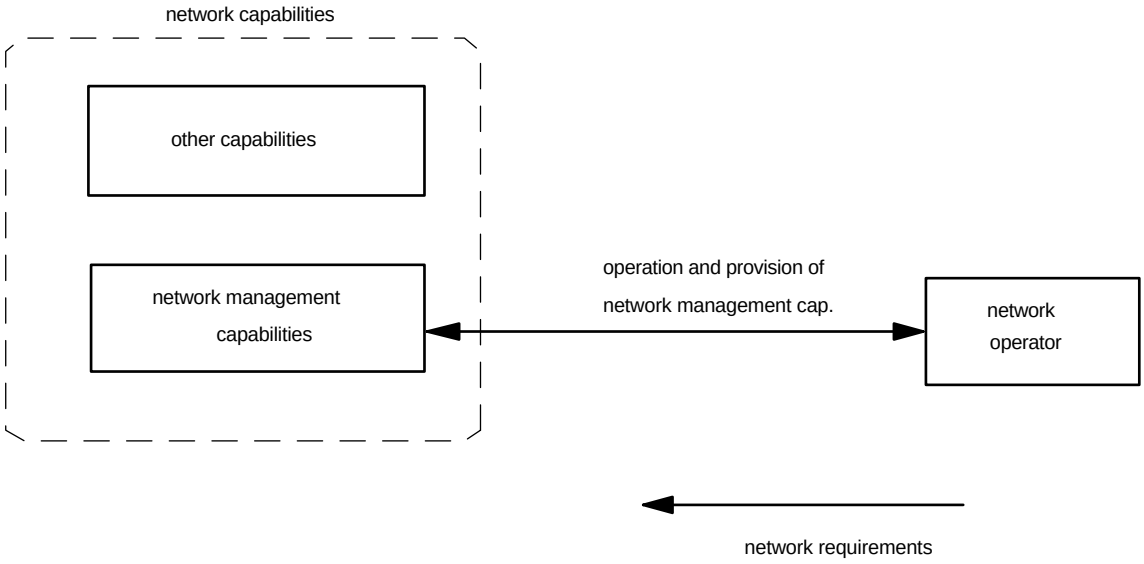


Figure 10: Network requirements for network management

The IN and non-IN requirements of network management are not essentially very different. The TMN application functions are relevant to IN as much as they are to non-IN. So, the following organisation of management capabilities (Annex B to Recommendation M.2x) are applicable.

Performance management

- Performance monitoring
- Traffic management and network management
- Quality of service observations

Fault (maintenance) management

- Alarm surveillance
- Failure localisation
- Testing

Configuration management

- Provisioning
- Status and control
- Installation

Accounting management

Security management

Additional application functions may be needed to deal with the IN situation; this is for further study. Also, additional requirements may exist in the case of IN, warranting further investigation of new objects and their operations on these objects; e.g. trigger table management, service logic data management, etc. This is for further study.

2.3.5 Service processing

Network requirements for service processing refer to the network capabilities that are necessary for the provision, from a network operator point of view, of basic and supplementary services by an IN-structured network. This is schematically shown in figure 11.

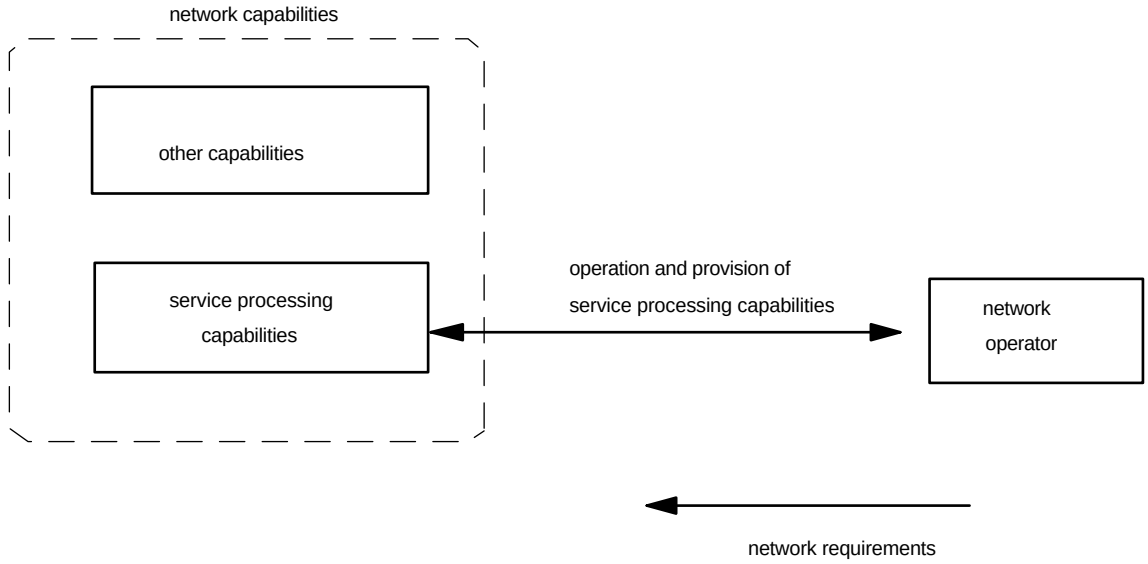


Figure 11: Network requirements for service processing

The main network requirements for service processing stem from the inability of network operators of traditional "non-IN" networks to rapidly create and deploy new supplementary services. To overcome this inability the IN aims for:

- fast service implementations by means of reusable network functions;
- modularisation of network functions;
- standardised communication between network functions via service independent interfaces.

To better understand how the IN can achieve the goal of fast service implementation, an overall comparison is made between traditional non-IN service processing (not meeting the aims listed above) and newly required IN service processing (meeting the aims listed above). Both cases are respectively referred to as the non-IN service processing model and the IN service processing model.

Non-IN service processing model

A simplified but typical representation of a non-IN service processing model is illustrated in figure 12.

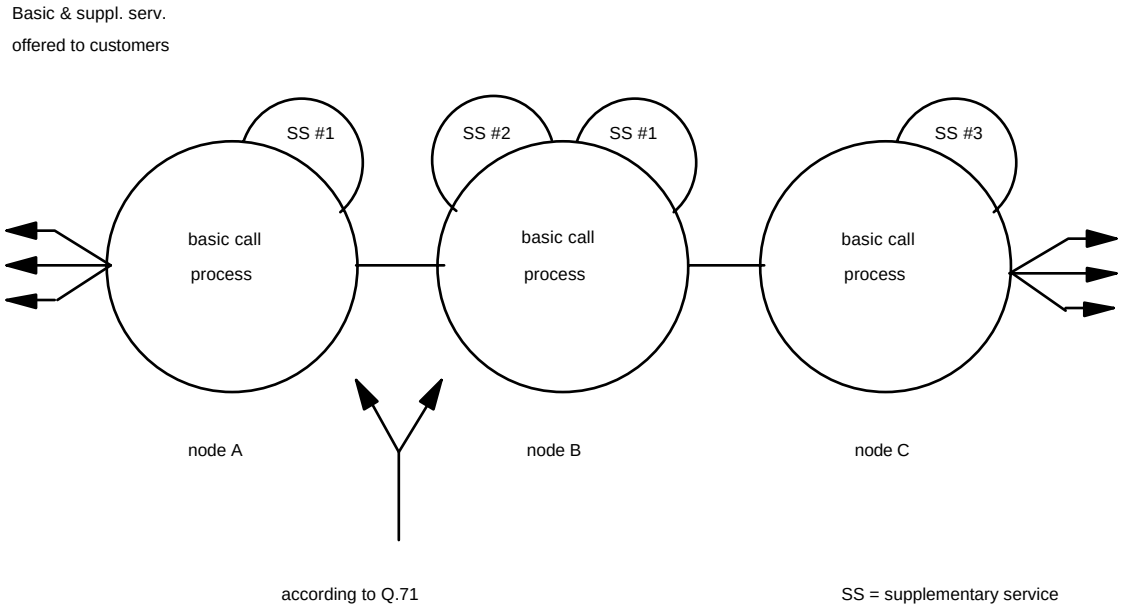


Figure 12: Non-IN service processing model

The main elements in this model are the basic call processes (in each node) and the extensions on these processes for supplementary services. The model is derived from and consistent with CCITT Recommendation Q.65 ("Stage 2 of the method for the characterisation of services supported by an ISDN") and CCITT Recommendation Q.71 ("ISDN 64 kbit/s circuit mode switched bearer service"). Some important characteristics of the non-IN service processing model are the following:

- the basic call process supports the provision of a particular basic service and is designed for optimal quality of service and network performance;
- each node contains an instance of the basic call process;
- in each node the basic call process can be modelled as a finite state machine that communicates with "neighbour" basic call processes;
- each supplementary service extension is a monolithic non-reusable entity that modifies or supplements the basic call process;
- there is no reusability of service-independent network functions;
- there is no fast service implementation.

For each introduction of a new supplementary service the basic call process (CCITT Recommendation Q.71) needs to be modified/supplemented. Generally this is a very time-consuming process because each new service requires new software generics to be loaded on the individual nodes. As a conclusion the non-IN service processing model does not allow fast service implementation.

IN service processing model

A high-level overview of a desirable IN service processing model is illustrated in figure 13.

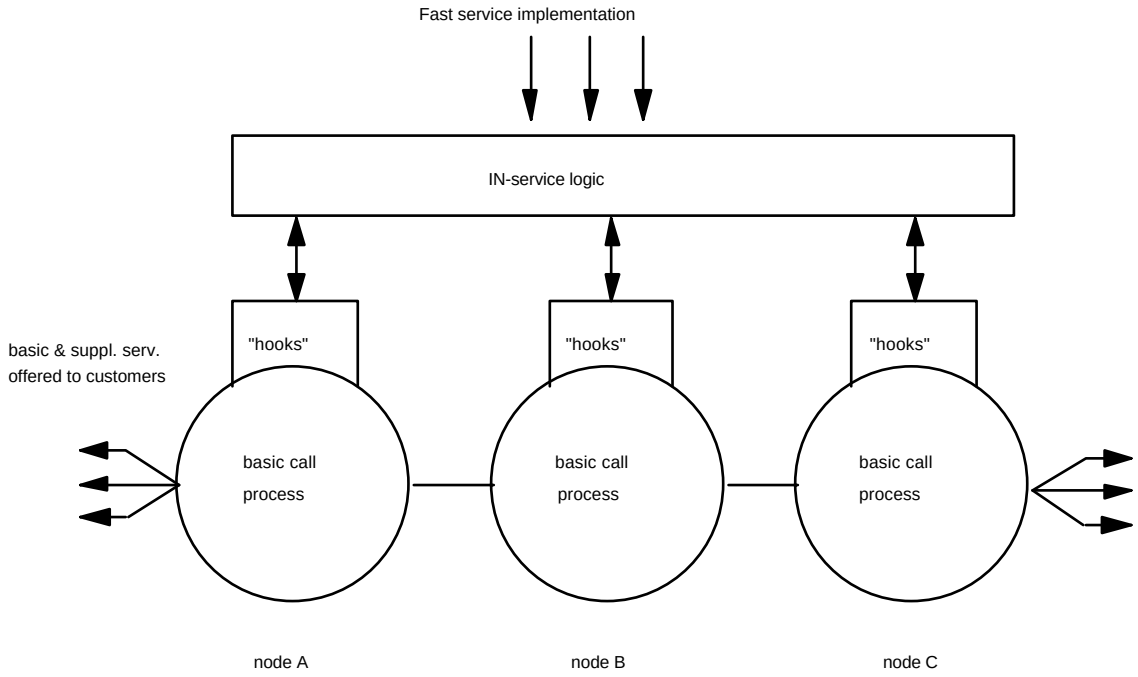


Figure 13: IN service processing model

The main three elements of this model are: the basic call processes, the "hooks" that allow the basic call processes to interact with IN service logic, and IN service logic that can be "programmed" to implement new supplementary services. For these elements the main principles are described below:

- The basic call process should be available all over the network and is designed to support, with optimal performance, services that do not require special features. In order to achieve flexibility in service processing, the basic call process needs to be modularised into service-independent sub-processes such that these can be executed autonomously (without interference from the outside during execution);

- "Hooks" are to be added to the basic call process forming the links between the individual basic call sub-processes and the service logic. The "hooks" are able to start an interaction session with the IN service logic. For this it should continuously check the basic call process for the occurrence of conditions on which an interaction session with IN service logic should be started. During an interaction session the basic call process is to be temporarily suspended;

- IN Service logic is a programmable software environment that needs to be developed to allow fast implementation of new supplementary services. New supplementary services can be created by means of "programs" containing IN service logic. The IN service logic is able, via the "hooks" functionality, to interact with the basic call process. In this way IN service logic can control the sub-processes in the basic call process and the sequencing of these sub-processes. IN addition IN service logic can decide to terminate an interaction session with the basic call process. The basic call process will then resume its execution as specified by the IN service logic. In order to allow fast service implementation, the IN service logic should have a logical view of the network resources that constitute the basic call process and additional (specialised) network functions.

Consequences for call modelling in an IN architecture

As part of the activities to define an IN architecture including the network elements within this architecture there is a need for a call model that describes the real-time behaviour of call control capabilities for the provision of basic & supplementary services. In order to be consistent with the principles of the above-described IN service processing model, the IN call model should cover the following aspects:

- it should specify which basic services can be supported by the model;
- it should model the basic call processes (each individual basic service may require its own IN basic call process);
- it should describe trigger mechanisms ("hooks") that allow the IN basic call process to interact with service logic;
- it should provide a logical view (from service logic point of view) of call processing functions and network resources, in this way allowing fast service implementation;
- it should specify the mechanisms according to which an IN basic call process may interact with the service logic (e.g. single-ended interactions, simultaneous interactions, service-logic initiated interactions, etc.);
- it should be evolvable from the existing technology base.

2.3.6 Network interworking

Network interworking is a process in which several networks (IN-IN or IN-non IN) cooperate to provide a service.

The need for network interworking capabilities results from the fact that customers may want to access services which span multiple networks. These networks may have different access types (PSTN, ISDN, etc.) and may have different levels of IN structuring (full, partial or no-IN structuring). Irrespective of the access type and of the level of IN structuring services should be provided to customers in a consistent way.

Network interworking requirements exist at different levels:

- service processing;
- service management;
- service creation;

and in each level, some network interworking related gateway functions need to be defined. The specific capabilities of the gateway functions described in the following subclauses are for further study.

2.3.6.1 Gateway functions for service processing

Figure 14 represents the different gateway functions required to support network interworking at the level of service processing, when two IN-structured networks cooperate to provide a service.

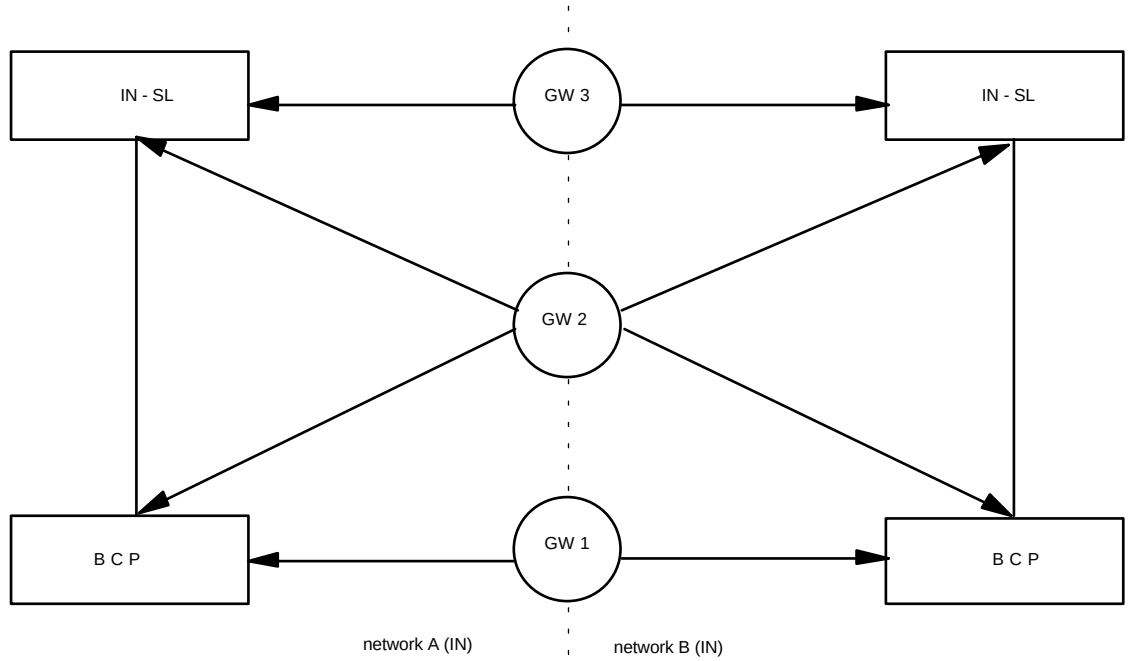


Figure 14: Gateway functions between two IN's

A gateway function may be used to access service logic in other networks (GW2), or to provide communication between pieces of service logic pertaining to different networks (GW3).

Figure 15 represents the use of interworking between in IN and a non-IN structured network.

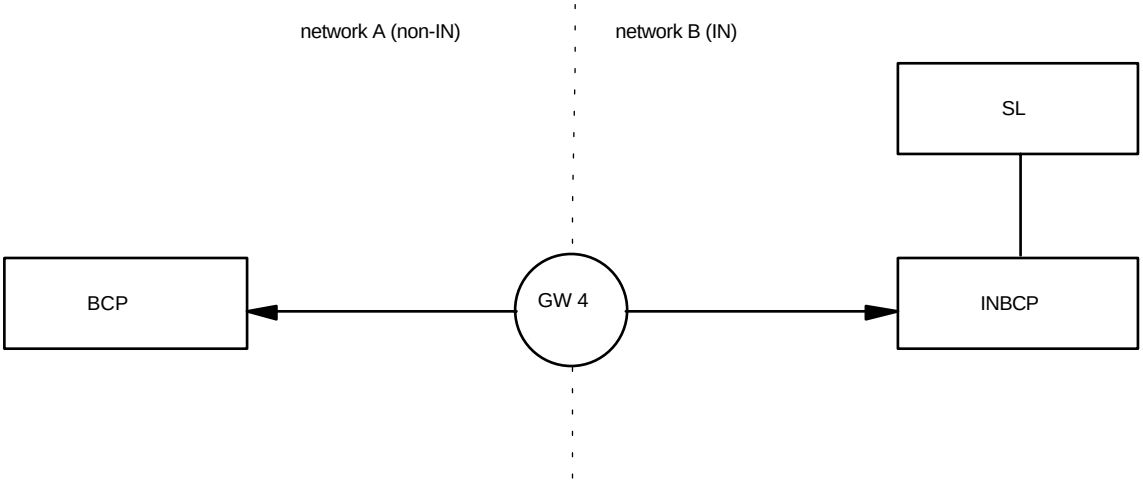


Figure 15: Gateway functions between IN and non-IN

A gateway function (GW4) is required to route the call between the IN and non-IN networks and to provide interworking between the BCP of network A and the BCP of network B.

2.3.6.2 Gateway functions for service management

At this level, a gateway function is required to link the service management function of the different networks which interwork. The gateway function should allow Service Deployment, Service Provisioning, Service Control Management, Billing and Service Monitoring for services that span multiple networks.

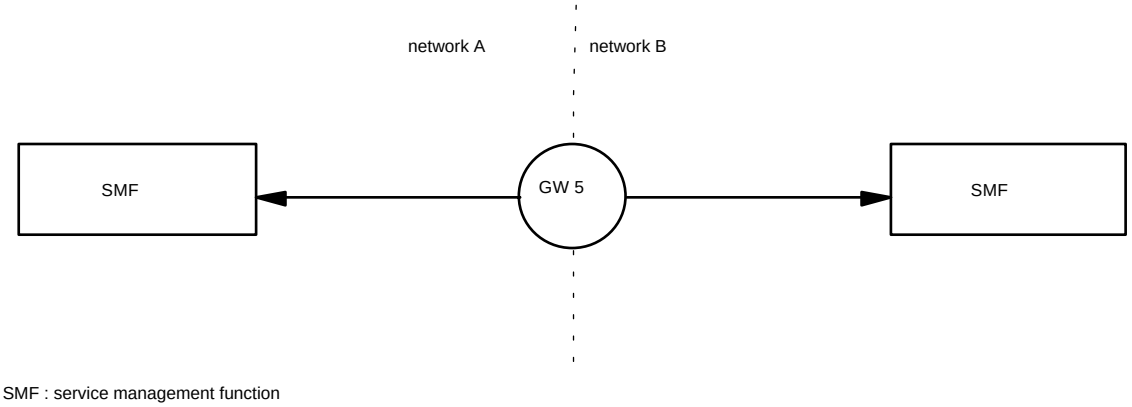


Figure 16: Gateway function for service management

2.3.6.3 Gateway functions for service creation

Service creation is an off-line activity and normally exist in a single network. The need for network interworking at service creation level is f.f.s.

3 IN architecture

3.1 Requirements and assumptions

The IN architecture is derived from Clause 1, "objectives, scope and definition of IN" and from Clause 2, "IN functional requirements".

A key objective of the IN is to provide service independent functions that can be used as "building blocks" to construct a variety of services. This allows easy specification and design of new services.

A second key objective is network implementation independent provision of services. This objective aims to isolate the services from the way the service-independent functions are actually implemented in various physical networks, thus providing services that are completely independent of underlying physical network infrastructure(s).

The network implementation independence has these objectives:

- services use distributed network functions in various ways;
- services can span several networks and are independent of specific implementations in these networks;
- services are independent of technological developments and evolution in network infrastructure, so that physical networks can evolve without affecting existing service provision;
- physical elements in such a network can be procured from different vendors.

To achieve these requirements, services are composed of service-independent building blocks (SIBs) which are standardised service, network- and vendor-implementation independent. Each SIB will consist of a set of functions which will be distributed into more than one physical element. Each of these functions will be standardised to give a consistent view of the resource(s) and control, thereby achieving vendor-independence.

3.2 The IN conceptual model

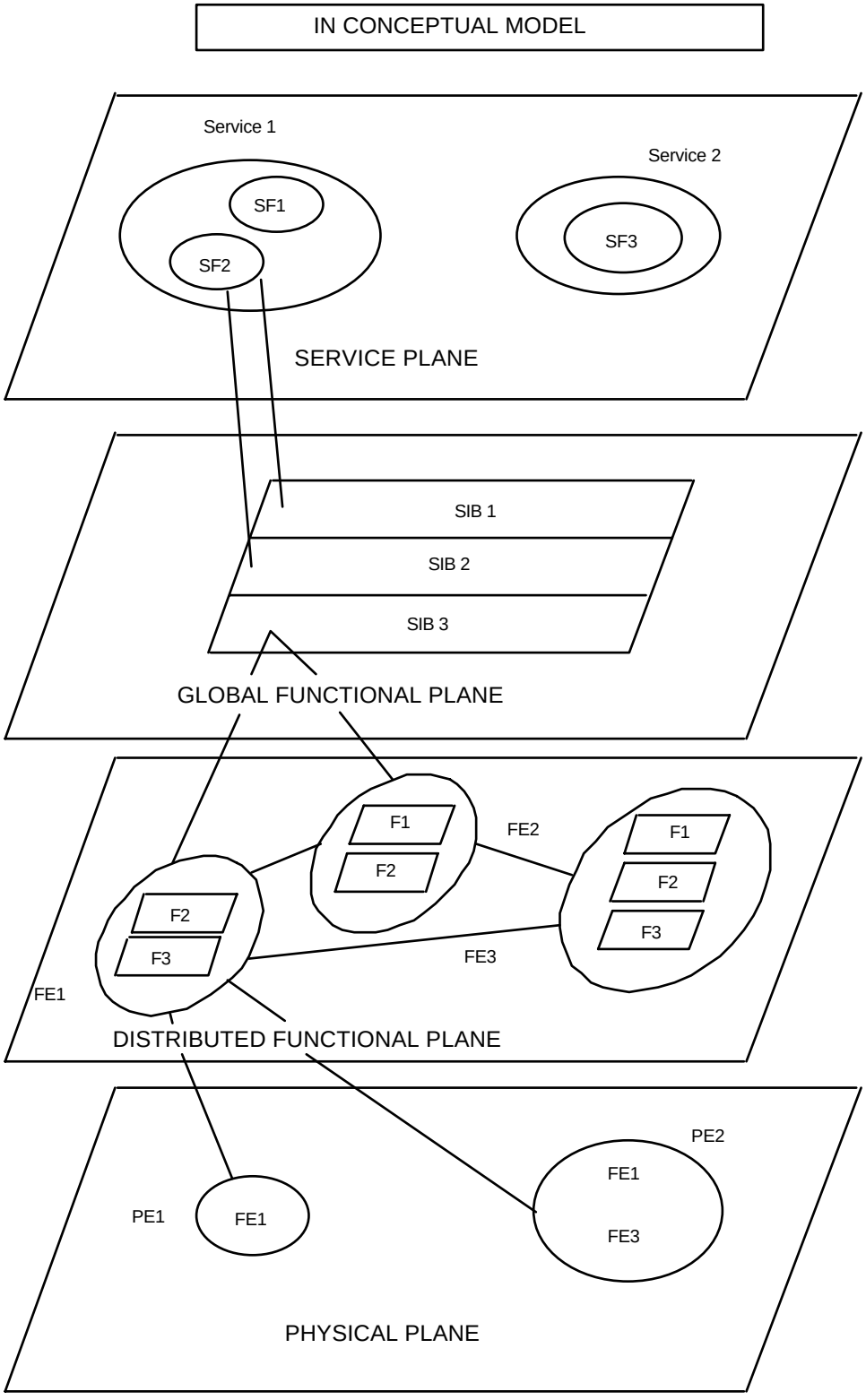


Figure 17: This figure illustrates basic aspects of the INCM

The IN conceptual model should not be considered in itself as an architecture. It is a modelling tool for the design of the IN architecture, taking into account Clause 1 guidelines, and in particular, the evolution of IN with its various phases.

Various "models" and "concepts" will be used in the standardisation of IN. The INCM is intended to represent an integrated, formal framework within which these concepts are identified, characterised and related. It should be possible to define clearly the purpose, value and limitation of any IN concept and its relationship to other such concepts. Existing concepts may need to be adapted for use within this framework. To achieve this, the INCM consists of four "planes" (figure 17) where each plane represents a different abstract view of the capabilities provided by an IN-structured network. These views address services aspects, global functionality, distributed functionality and physical aspects of an IN. IN only exists, as such, in the actual networks deployed. Because each plane represents a different view of the same model, there is no interaction among planes; however the entities in adjacent planes are related to each other.

3.2.1 Service Plane (SP)

The Service Plane represents an exclusively service-oriented view. This view contains no information whatsoever regarding the implementation of the services in the network, e.g. an "IN-type" implementation is not visible. All that is perceived is the network's service-related behaviour as seen, for example, by a service user. Services are composed of one or more Service Features (SFs), which are the "lowest level" of services.

3.2.2 Global Functional Plane (GFP)

The Global Functional Plane models an IN structured network as a single entity. Contained in this view are the Service Independent Building Blocks (SIBs).

3.2.3 Distributed Functional Plane (DFP)

The Distributed Functional Plan models a distributed view of an IN structured network. SIBs are composed of one or more standardised functions which are contained in this view. The DFP gives a static descriptive model by illustrating how functions may be aggregated into Functions Entities (FEs). Functions may be replicated in different FEs and each FE generally contains a subset of the total set of functions within the network.

It also gives a dynamic descriptive model of how the FEs may dynamically interact in a specific instance of execution of a service.

3.2.4 Physical plane

The Physical Plane models the physical aspects of IN-structured networks. The model identifies the different physical entities and protocols that may exist in real IN-structured networks. It also indicates which FEs are implemented in which physical entities.

3.2.5 Relationship with the 3 stage method

The I.130 based 3-stage method needs enhancing for IN. The correspondence between the 4 planes of the INCM and the 3 stage method is as follows:

- the service plane corresponds to the stage 1 description of a **service/service feature;**

- the GF plane has no correspondence to the (current) 3-stage method. It allows for describing services in an independent way, viewing the network as a single entity;

- the distributed functional plane is a service-independent stage 2 description and is applicable to an **IN functional architecture**;

- the physical plane corresponds to the stage 3 description.

3.2.6 Service logic

Service Logic may have different representations within each planes (see figure 18) e.g.:

- Global Functional plane: there is one set of logic per Service Feature and it uses SIBs;

- Distributed Functional plane: a set of service logic in the GF plane may be represented here by a set of service logic programs co-located with the SCF FE(s);

- Physical plane: everywhere the SCF FE is implemented, service logic programs may be loaded and executed.

3.2.7 Relationships among different planes

As noted in subclause 3.2, the entities contained in adjacent planes of the INCM are related to each other. The nature of the relationship is as follows:

- Service to GF plane: Service Features within the Service Plane are realised in the GF plane by a combination of Service Logic and SIBs and the global call model. This mapping is related to the service creation process;

- GF to Distributed Functional (DF) plane: each function identified in the GF plane must be present in at least one FE in the DF plane. A function may be replicated in more than one FE. Thus, co-operation of several FEs may be needed to realise a SIB. The service logic in the GF plane maps onto one or more SLPs in the DF plane. This mapping is related to the service creation process;

- DF to physical Plane: FEs identified in the DF plane determine the behaviour of the Physical Entities (PEs) onto which they are mapped. Each FE must be mapped onto one physical entity, but, each PE contains one or more FEs. Relationships between FEs, identified in the DF plane, are specified as protocols in the Physical Plane. SLPs may be dynamically loaded into physical entities and this mapping is related to the service management process.

3.2.8 Service interaction

A set of rules must be provided at the service plane level - these services may be "IN" or "non-IN". These interactions will also impact all the other planes. In addition to the set of rules at the service plane level, a robust mechanism for resolving easy service interaction should be provided in other planes in a service-independent manner to facilitate rapid service introduction.

3.2.9 Service and network interworking

Interworking on distributed functional plane

This plane should be explicitly divided into several parts, each representing one functional network.

The Interwork Function (IWF) should be identified between the functional networks

The service logic programs should reside in each functional network

Relationship between IWF and other intra-network FEs in the same network should be identified for network interworking capabilities

Information flow between IWFs in different network should be studied considering the network interworking capabilities, network security and network integrity.

Interworking on physical plane

This plane should be explicitly divided into several parts, each representing one physical network

The Inter-Work protocols should be identified at reference points between the physical networks, and be standardised

The service logic programs which reside in each functional network are loaded in physical entities in the respective network.

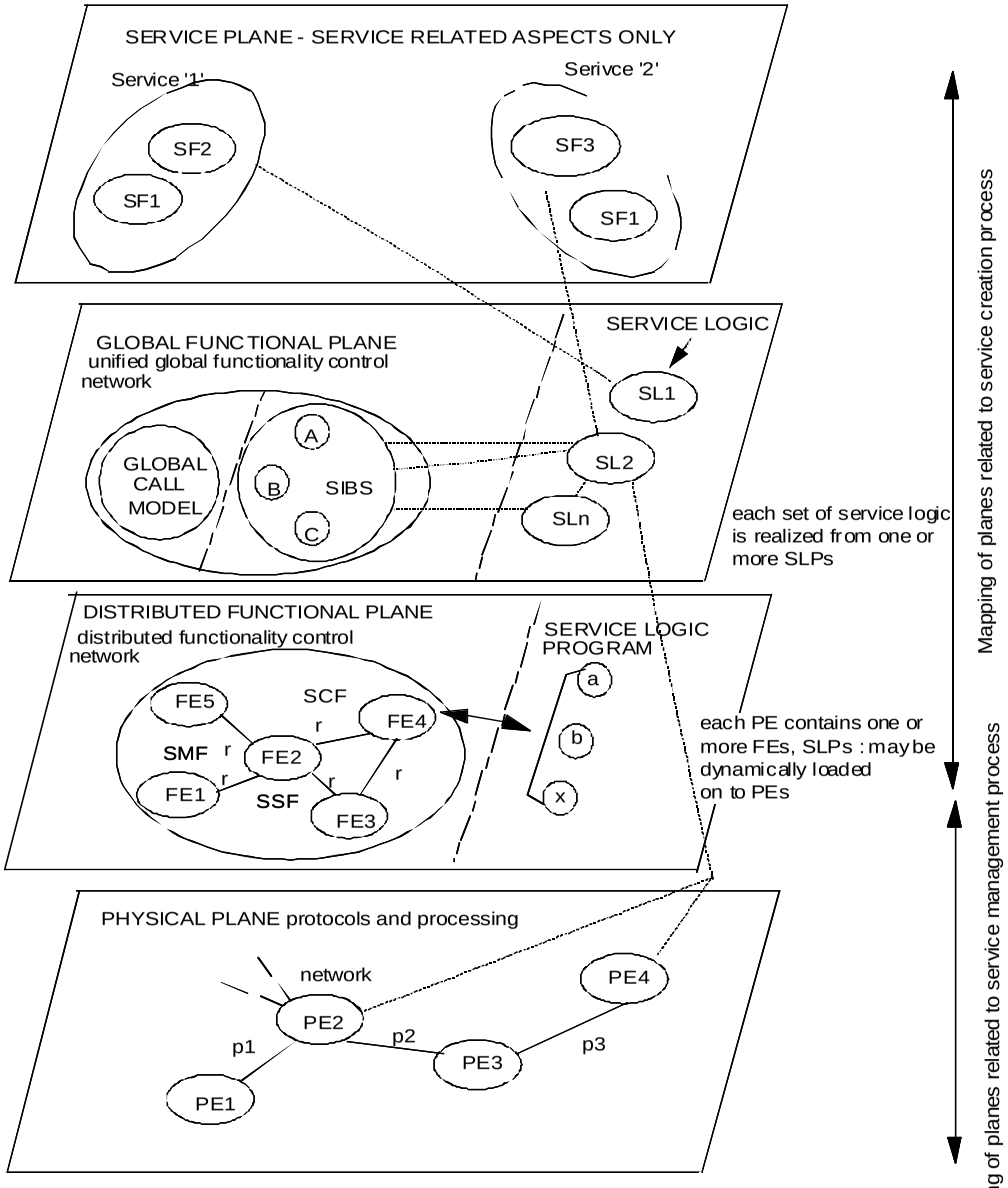
3.2.10 Management functionality

Management is related to ALL planes of the IN conceptual model. In an IN-structured network there is a need to consider both the service and network aspects of management. Specific text on these aspects in contained within each architectural subclause (3.4 to 3.7).

One particular aspect of management (e.g., service creation, service introduction, service tailoring, customer control ...) can be viewed as non real-time, and independent of the actual real-time service execution. In a multivendor environment, different versions of physical representations of the same functional entity may exist, where versions may contain sub-sets of capabilities of other versions. It is for further study how the conceptual model should reflect these non real-time dependencies.

3.3 OAM

The application of OAM to the IN Conceptual Model is for further study.



SF - service feature
SIB - service independent building block
FE - functional entity
PE - physical entity
SL - service logic
SLP - service logic program
SMF - service management function
SCF - service creation function
SSF - service switching function

Key :
service creation & management processes
r : relationship
pi : protocol

Service creation and management process and recognised as mapping of adjacent planes

Figure 18: IN conceptual model with service logic

3.4 Service plane architecture

3.4.1 General

The service plane illustrates that new services can be described to the end user by means of a set of generic blocks called "Service Features".

The service plane represents an exclusively service-oriented view of I.N. This view contains no information whatsoever regarding the implementation of the services in the network - all that is perceived is the network's service related behaviour as seen, for example, by a customer.

Furthermore, Management Services are contained in the Service Plane; they can be described to the end user by means of Service Management Features.

3.4.2 Characterisation of services and service capability requirements

Characterisation of services is to identify service independent capabilities that are required to construct and/or customise services by the customers or network operators. From the customer point of view, the service capabilities required are, e.g. call queuing, customised announcements, etc.

There is a need for a structured approach with which to classify service characteristics and identify service capabilities. The structured approach shown in figure 19 below demonstrates a methodology for analysing services decomposing services into service independent capabilities. These re-usable service independent capabilities (such as number translation, routing, or charging) will form the basis for input to service plane modelling, global functional plane modelling, distributed functional plane modelling and connection control modelling.

It is beneficial that activities involving functional modelling make use of the results of such service analysis, based on the characterisation of services for verification of their models, and to ensure a unified model for service processing.

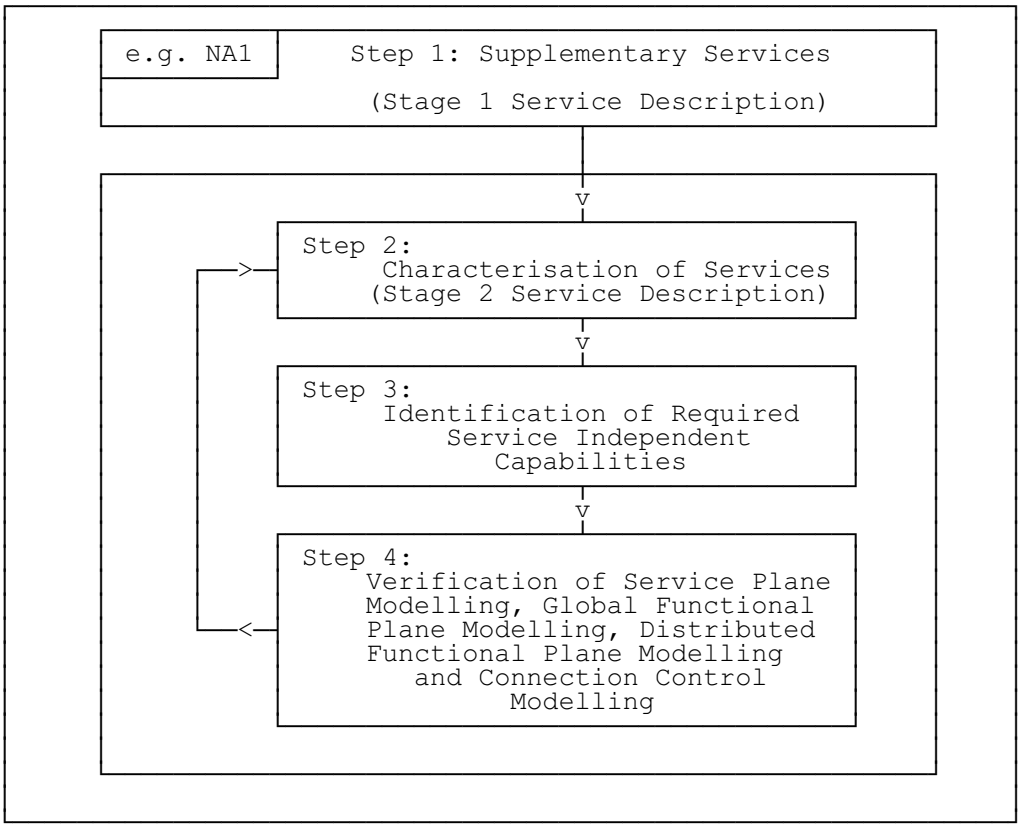


Figure 19: Methodology for service characterisation

Step 1: Existing service specifications as well as emerging supplementary service specifications (e.g. from NA1) are selected as candidates for analysis.

Step 2: These services are characterised in a certain way in order to identify common capabilities in the service.

Step 3: The analysis in step 2 results in requirements in the form of service independent capabilities which are input to Service Plane Modelling, Global Functional Plane Modelling, Distributed Functional Plane Modelling and Connection Control Modelling.

Step 4: Service Plane Modelling, Global Functional Plane Modelling, Distributed Functional Plane Modelling and Connection Control Modelling are verified against the required service independent capabilities (step 3) from service characterisation (step 2).

3.4.3 Service interaction

This subclause focuses on the interaction between supplementary services and between basic services. The service interactions are described from the user point of view.

Service interaction applies to all interactions of the service being described with other services which have been identified.

An IN structured network shall be able to handle multiple services for the same call. The necessary interactions shall be defined for the processing of several services for the same call. Where multiple services can be activated concurrently some prioritisation of services will be necessary. Additionally, certain services may override or deactivate other services. Aspects such as user specific service request taking priority over group service requests should also be possible.

The service interaction is part of the specification of services, and should be dealt with in the service plane modelling. There are often many ways to deal with an interaction between two or more services. In an IN structured network, service interactions may be customised.

How service interactions are implemented is not visible in the service plane. The usage of the service independent mechanism in the IN architecture to handle service interactions will be visible in the Global Functional, the Distributed Functional and the Physical Planes.

Regarding service interactions, the following issues need consideration:

- at different phases of a call, i.e., originating, terminating, interrupt (active) and release phase of call processing;

- when a service spans over more than one network. They may impose additional requirement on service interaction, which need further study;

- service interaction may occur between services on behalf of a single customer, as well as between services on different customers.

Example of service interactions are given below:

- Abbreviated Dialling and Number Screening;
- Freephone and Call Forwarding Unconditional;
- CLIR and CLIP;
- Call Forwarding and Premium Rate Service;
- Call Waiting and Call Forwarding Busy;
- Conference Call and CUG;
- Meet-me Conference and CUG.

Examples of different interactions between freephone and call forwarding unconditional are:

[1] freephone calls shall not be forwarded, but the selected destinations shall be called;

[2] freephone calls shall be forwarded like other terminating calls;

[3] a freephone destination, which has activated call forwarding unconditional, shall not be selected for freephone calls.

3.4.4 Service plane modelling

Service comprise one or more service features (SF). Service features could represent a completely self-contained service. SFs comprises one or more SIBs.

All individual telecommunications services identified in the service plane should be described as seen from the users viewpoint using stage 1 service definition methodology given in CCITT Recommendation I.130.

It is noted that a network operator can make use of Telecommunications services provided by his or another network to support his operational requirements; it is noted that Management Services can be used by the network operator (e.g. service deployment) and by the subscriber (e.g. customer control).

3.4.5 Mapping of requirements

3.5 Global functional plane architecture

3.5.1 General

The global functional plane models an IN structured network as a single entity.

Contained in this view are the service building blocs (SIBs) which are performed in a network to provide services to customers.

SIBs are meaningful units of network functions used to design services and service features. Furthermore, SIBs can be used to design management services.

Detailed implementation of the SIBs should be hidden and it should present simple external interfaces to the designer of the services.

SIBs should be of a global nature and their locations need not to be considered as the whole network is regarded as a single entity.

SIBs are defined from the point of view of the designer of the service while Service features are defined from the point of view of the Telecom end-user.

"Attributes" used to identify SIBs are:

- service independence, network implementation independence, vendor implementation independence;
- residence is the SIB plane;
- service logics use multiple SIBs;
- the user of the SIB is the designer of the service;
- a SIB has one or multiple inputs and one or multiple outputs;
- SIBs need to be administered in their life cycle when implementation aspects are considered because implementation may evolve. However, the interface to the service designer remains stable;
- a SIB could access/manipulate abstract resources/objects in a single network.

All functional requirements for service independence are identified at the SIB plane level. These requirements are in the form of SIBs.

SIBs are service independent in the sense that:

- a SIB may be used by several services;
- the definition of a SIB should be stable;
- a SIB does not know for which service it runs (parameters may be used for each specific usage);
- the list of SIBs should be stable- SIBs are reusable; they are used without modification for other services;
- SIBs are easy to use.

Mapping from the service plane to the SIBs plane is like programming a service on a "virtual machine". This process is the decomposition of services into SIBs. Both the process of combining these building blocks ("service creation") and the ability for the end-users to customise service attributes ("customer control") are considered management activities.

For the description of SIBs, functions should be used (not EFs so as to avoid confusion with the definition in the I-series of recommendations) and they are related to FEAs in the DF plane.

The service logic denotes all the service specific logic. Services are specified with service logics which will be produced via the service creation environment. Service logic uses SIBs which are realised within the operational network as combination of Functional Entity Actions (FEAs). Service logic is specified on top of a global view of the network (point A in figure) in the sense that it ignores the actual configuration of the network. Thus, services see the network as a single virtual machine.

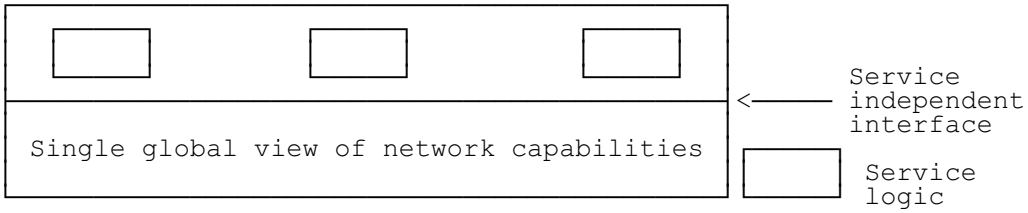


Figure 20: Programming type of interface

General open points:

[1] Are SIBs sets of operations (00 approach) or single operations (functional approach) ?

[2] May a SIB invoke another SIB ?

[3] How are we going to hook them together ?

[4] Is there some control decision embedded in the SIB ?

[5] What is the relationship of service logic to SIBs ?

3.5.2 Global Functional Plane modelling

3.5.2.1 Identification of SIBs

One useful approach to identifying SIBs is to divide the existing call processing into phases. This approach enables a systematic decomposition of services into functions. The phases of generic call processing together with example function are:

Originating phase

Call processing is the phase during which such functions as message waiting indications, service denial, route list restriction, predefined routing, address to route translation, service request reception or expensive route warning take place.

Termination phase

Call processing is the phase during which such functions as route to alternate destination (call forwarding, hunting, distribution), delivery of call related informations (e.g. caller identification, etc.), alerting including alternative alerting under defined conditions, or queuing take place.

Interrupt phase

Call processing is the phase during which such functions as accepting service requests from the user while the call is in active state, or adding and deleting users from a call take place.

Release phase

Call processing is the phase during which such functions as provision of billing estimates for the call, or other functions made necessary by pending requests takes place.

Analysis of telecommunication services is another useful approach for determining the functions that must be provided by the network to support these services. In examining similarities among these service-oriented functions, redundancies become apparent. As a result, a set of SIBs could be constructed to include all the required functions to support telecommunication services. Such a set would contain a sufficient number of SIBs to be used in various combinations to support all required telecommunication services.As an example of service analysis figure 21 illustrates a decomposition from the Service Plane to the SIB plane; table 1 shows this decomposition in more detail.

Tables 2 and 3 then illustrates a possible method of continuing the analysis into the DF plane mapping the SIBs to the Functional Entity Actions, and then mapping the FEAs to the Functional Entities.

NOTE: Service, Service Features and FEAs used in the above mentioned figure and tables are only examples.

Another approach to identifying SIBs is to identify service independent extensions to IN basic call processing, starting from the analysis of the relationship between the Functional Entities needed to support a specific service. Note that service independent extensions (SIEs) are not SIBs, but they form part of SIBs. The process follows a two-step sequence:

[1] Service analysis by means of functional decomposition. This corresponds to step 2 of the methodology for service analysis to identify the common capabilities in the service (refer to figure 19 in subclause 3.4.2). These common capabilities are identified and described by means of Information Flows. This is analogous to Step 2 of the Stage 2 description method for the characterisation of services supported by an ISDN (CCITT Recommendation Q.65).

[2] Service independent extensions (SIEs) to IN basic call processing are identified. This corresponds to step 3 of the methodology for service analysis to identify the required service independent capabilities as shown in figure 19 of subclause 3.4.2. The SIEs are identified by way of Specification Description Language (SDL) diagrams. This is analogous to Step 3 of the Stage 2 description method of CCITT Recommendation Q.65.

As an example the Freephone service is used; analysis of the Call handling/Routing functions is provided by means of information flows between the different FEs involved in the service according to the step 2 of the stage 2 of the method for the characterisation of services supported by an ISDN (CCITT Recommendation Q.65). For each FE, the ISDN basic call state and the FE actions are indicated to illustrate the relationship between the network function and the extensions to the basic call model. These extensions are described using SDL diagrams.

The Call Handling/Routing analysis addresses only the initial call setup portion of the basic call model, between the IDLE and the CALL SEND states, within which the routing function is provided.

- figure 22 shows the Call Model relationship.

- figure 23 gives the information flows diagram, showing the actions performed by the originating LE.

- figure 24 gives the SDL diagram of ISDN basic call processing in the SSF; the shaded area represents the required extension to ISDN call processing to be performed by the SSF.

- figure 25 gives the SDL diagram of the ISDN basic call processing extension for call routing needed in the SCF.

3.5.2.2 Classification of SIBs

The SIBs should provide the modularity within the SIB plane that is required by the definition and objective of the IN concept; the following groups (or families) of SIBs are currently identified:

Translation

This group contains SIBs used to translate an information into a different one.

User interaction

This group contains SIBs used to interact with the user, e.g. for sending information, receiving information and prompting to and collecting from the user.

Call control

This group contains SIBs used to manipulate connection and disconnection; these SIBs also deals with some end-to-end bearer control; multiparty and multimedia call are handled by attributes.

Security

This group contains SIBs used for authentication, authorisation, screening and validation (syntax verification).

Management

This group contains SIBs used for managing, collecting, retrieving, updating information; for defining activation/deactivation/condition criteria of service logic invocation ...

Monitoring

This group contains SIBs used to monitor events such as release of a call, state of a user access ...

Charging

This group contains SIBs used to charge the call (e.g. creating a billing records or sending charging pulses to the caller, etc.).

3.5.2.3 SIBs description methodology

The SIBs should provide the modularity within the SIB plane that is required by the definition and objectives of the I.N. concept. In order to effectively progress such studies, however, a method is required to characterise and technically describe the SIBs.

Techniques analogous to those used in the three stage service definition methodology (I.130), i.e., prose description, static description, and dynamic description, are appropriate, taking into account that the CCITT Recommendation I.130 is a "service and network dependent" methodology.

In order to use the SIB in the combinations necessary to support telecommunication services, analysis of their respective roles on the SIB plane, and their relationships to services (Service Plane) and FEAs (DF plane) is required.

A proposed SIB description methodology is as follows:

Service designer description of SIBs

Definition: This subclause provides a short description of the SIB from a service creation point of view.

Description: This subclause expands the definition and describes the SIB in term of inputs, operations and outputs from a service designer stand-point.

Parameters: This subclause describes in detail the internal, input and output parameters necessary to identify the SIB. Each parameter will be described and an indication if mandatory or optional will be inserted.

Irregularities: This subclause describes the cases of unsuccessful outcome of the SIB. The detailed description of the causes or error will be inserted.

Interaction with other SIBs: This subclause describes, when applicable, the interactions of the present SIB with other SIBs running in the same service.

Flow Diagram: This subclause is a graphical representation of the SIB from a service designer stand point using SDL macro diagrams (CCITT Recommendations Z.100 and 104).

Functional description of SIBs

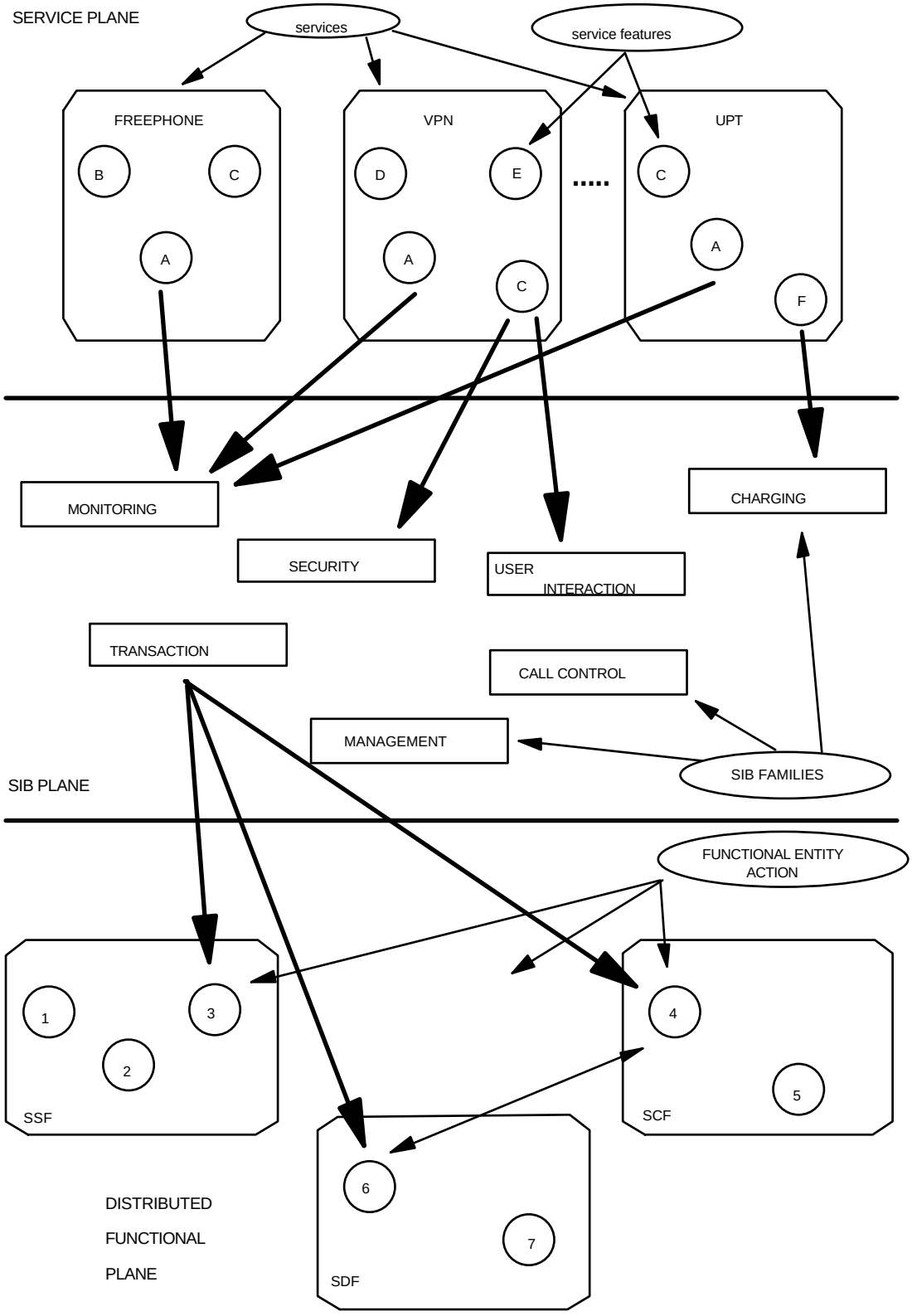


Figure 21: Service decomposition

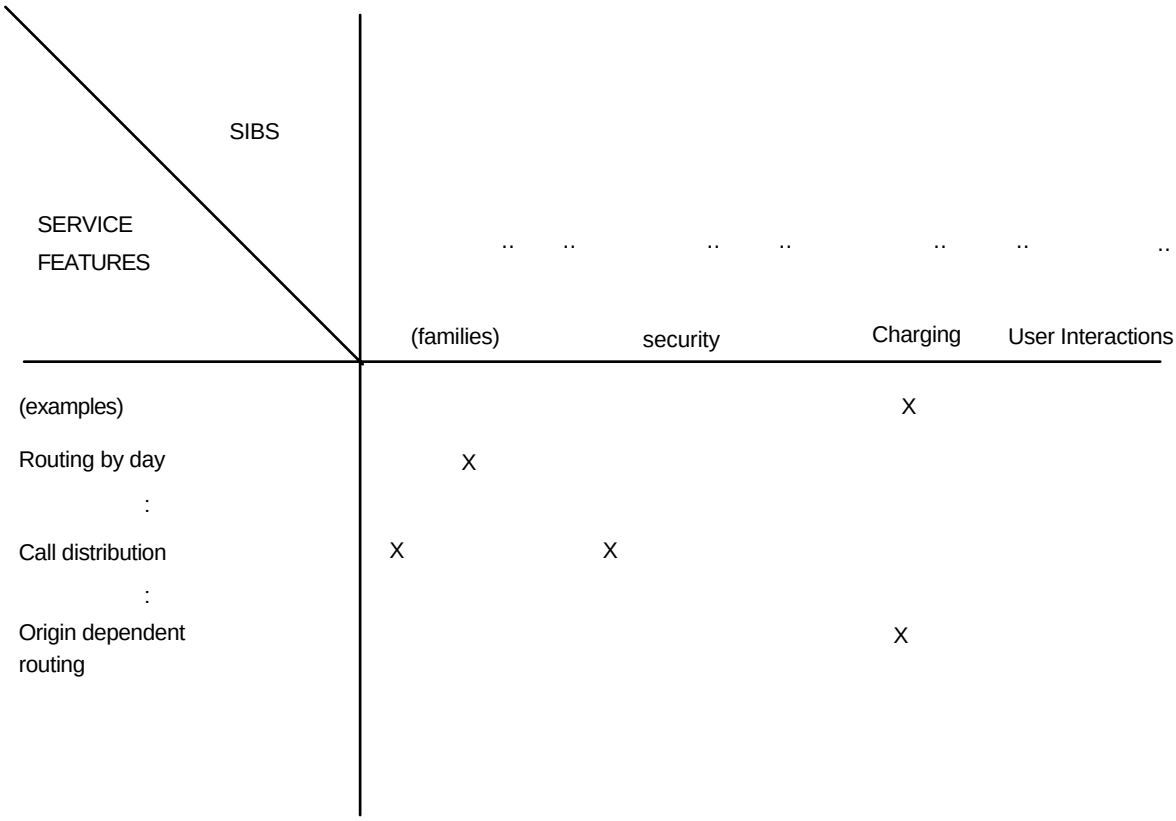


Table 1: Decomposition of service features to SIBs

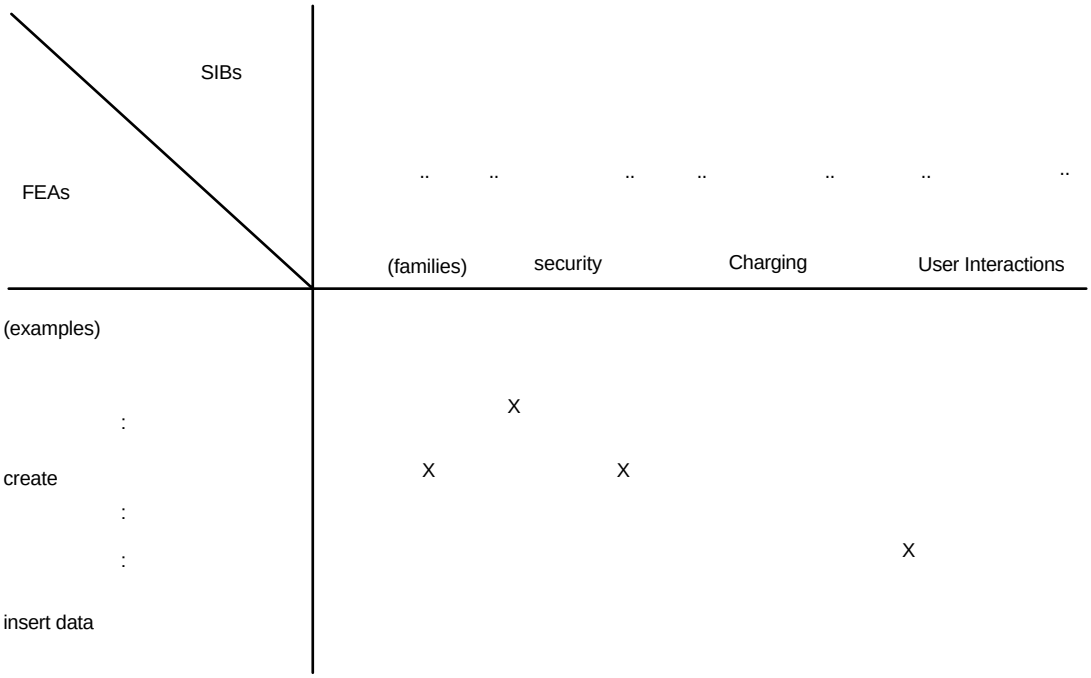


Table 2: Decomposition of SIBs to FEAs

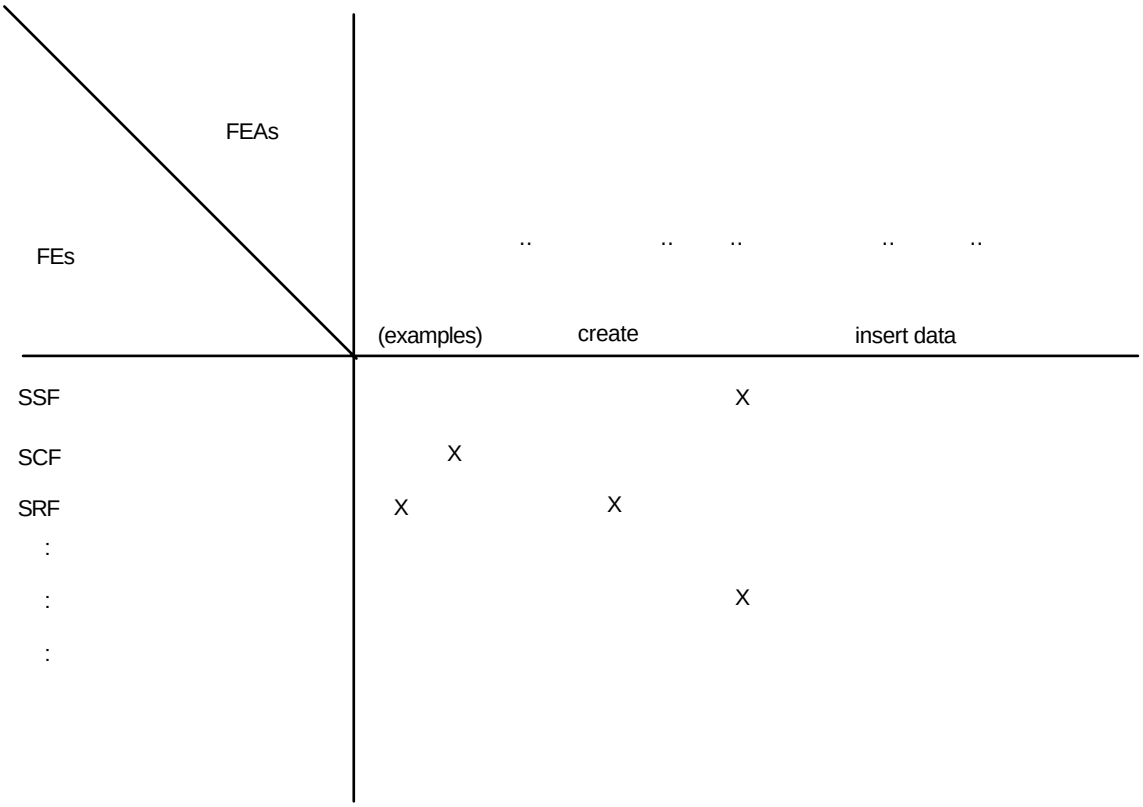


Table 3: Mapping of FEAs to FEs

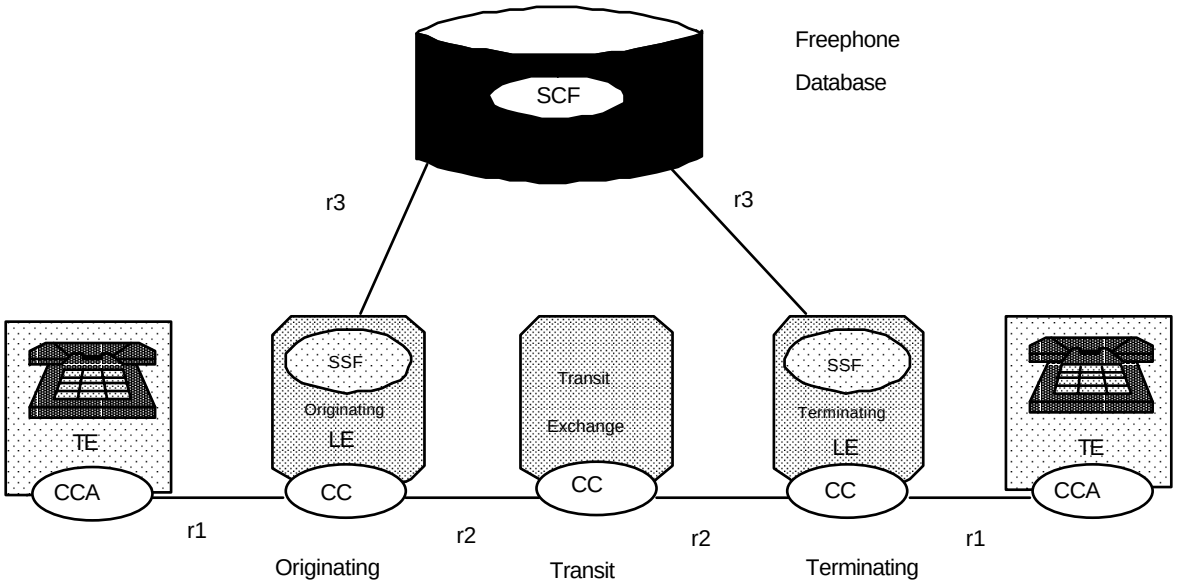


Figure 22: Freephone FEs to ISDN basic call model relationship

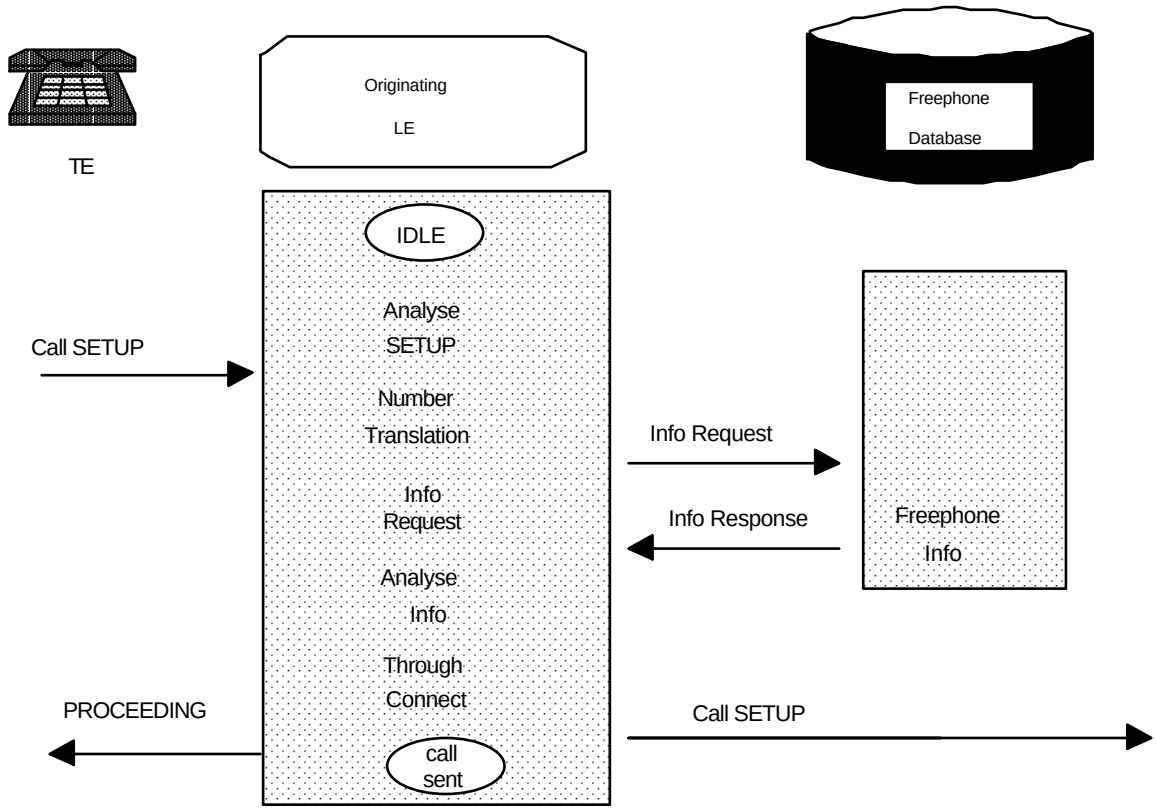


Figure 23: Freephone call handling/routing

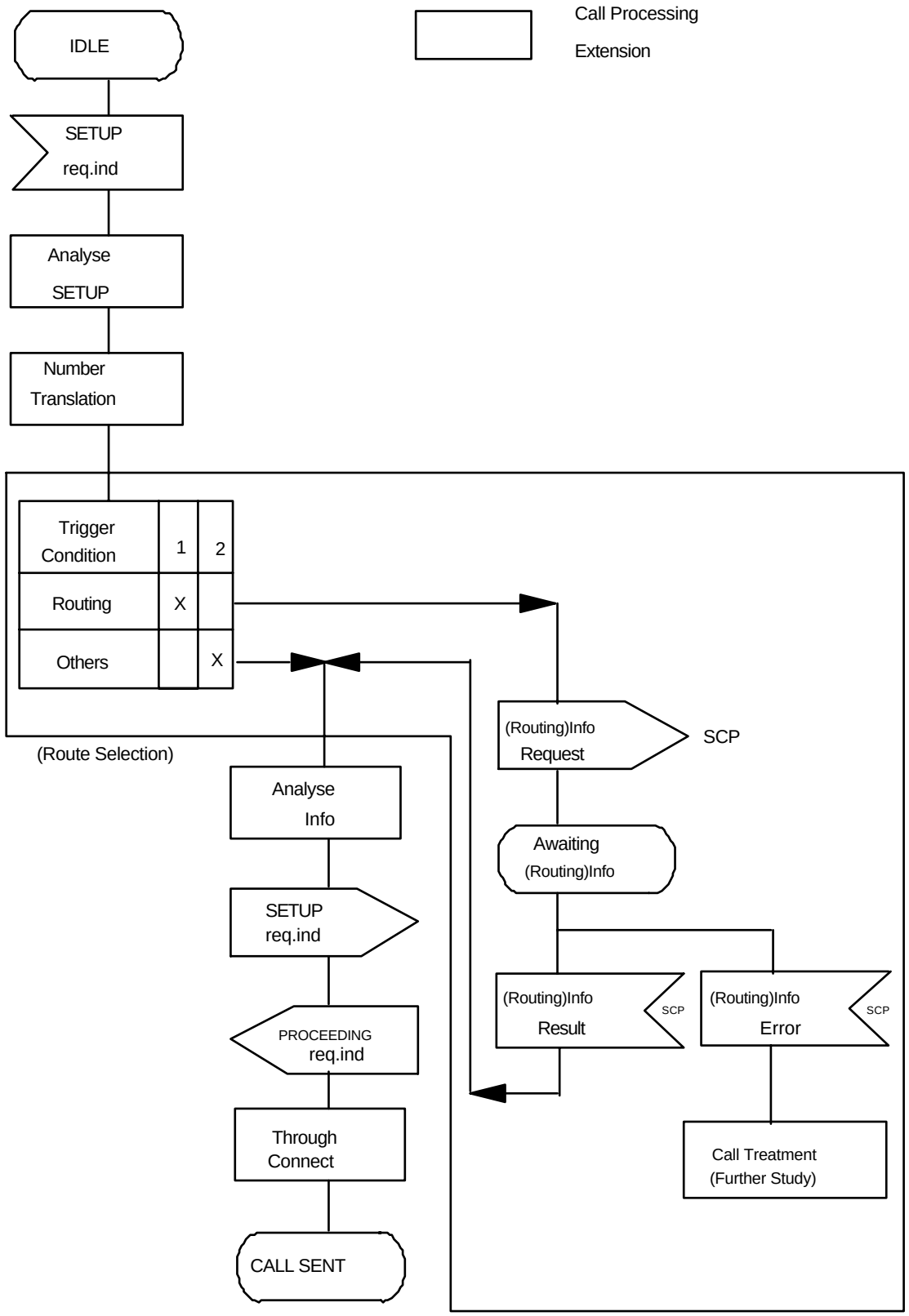


Figure 24: SSP/SDL diagram for call handling/routing

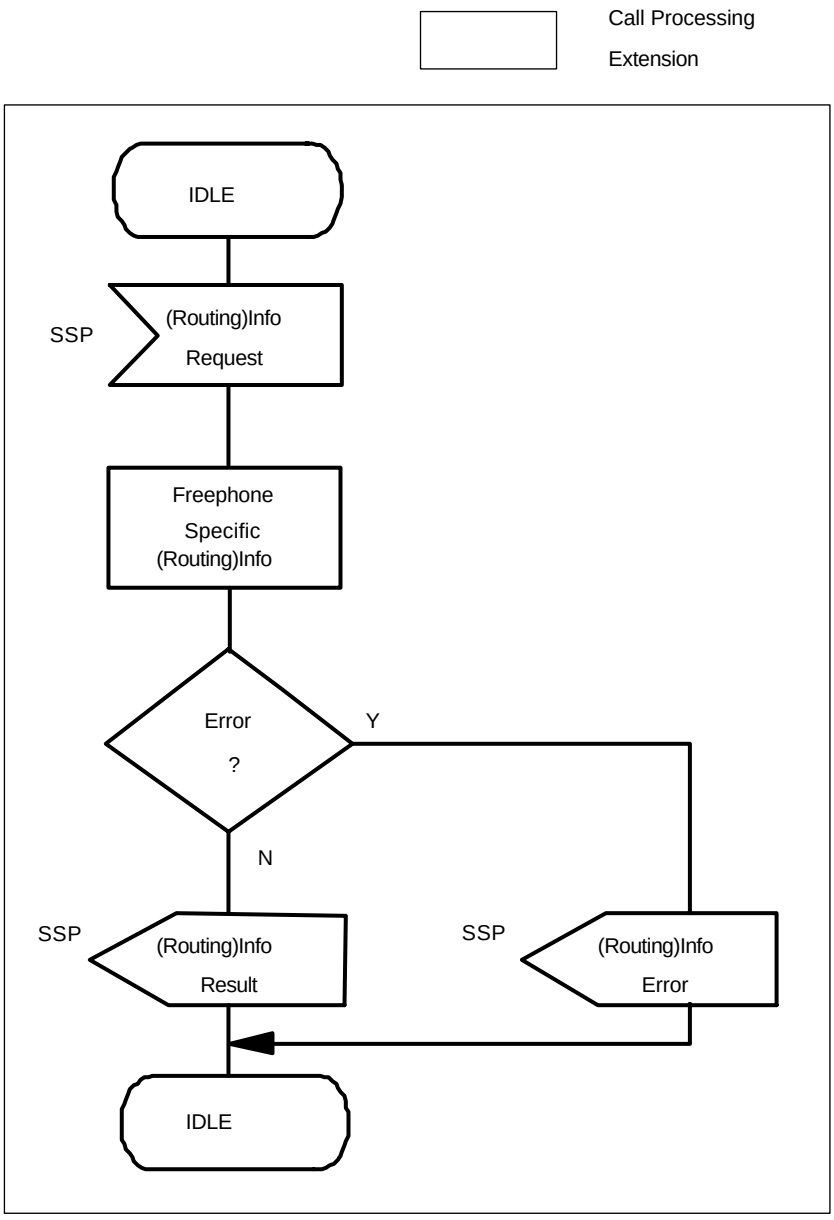


Figure 25: SCP/SDL diagram for call routing

NOTE 1: Following mechanisms have been identified:

mechanism to resolve service priority

mechanism to determine which service has priority to access certain resources

need to resolve multiple concurrent service invocations

when a particular service is already active and another is activated need to ensure that the additional service has no detrimental effects on the previous one

use of the trigger check points may resolve certain aspects of service interaction

NOTE 2: There is also need to develop general rules for service interaction, for example:

user specific requests take priority over group requests

certain services override others

3.5.3 Mapping of requirements

3.6 Distributed functional plane architecture

3.6.1 Introduction

3.6.1.1 Requirements and assumptions

The DFP architecture is consistent with the framework defined by the IN conceptual model:

- it identifies the specific elements and the relationships between them that are necessary to support the objectives of I.N.;

- the DFP architecture models the functions to be performed in I.N. structured network.

The DFP architecture should be service independent to provide the flexibility to support a large variety of services and to aid the evolution of I.N. by organising the set of functions in an open-ended and modular structure.

The DFP architecture should be vendor/implementation independent, thereby providing the flexibility for multiple physical networking configurations, and placing no constraint on national network architectures beyond the network and interface standards which will be developed for IN structured networks.

The DFP architecture should be sufficiently complete in the definition, including e.g.: OA & M, to be used as input to derive recommendations on protocols and procedures between physical elements, and should offer substantial implementation flexibility to administrations and manufacturers.

The methodology used to describe the DFP architecture should allow for:

[1] a precise definition of functional capabilities and their possible distribution in network equipments;

[2] a detailed description of what functions and information flows are to be provided, but not how they are to be implemented;

[3] a single functional specification which can be applied in a number of physical realisations.

3.6.1.2 Definitions

What follows describes the methodology (based on CCITT Recommendation Q.65) being used for the DF plane.

A FUNCTIONAL ENTITY is a grouping of Service providing functions located in the same physical device.

This excludes utility/Housekeeping functions which are not directly involved in providing a service. This explains why there is no FE identified to describe the communication between FEs.

FEs are assigned unique FE IDENTIFIERS.

One FE cannot be split into different physical entities. Furthermore, when functions need to reside in different units of equipments, they shall be assigned to different FEs.

The physical location of functions is not the only criteria for grouping functions into an FE. FE grouping criteria should take into consideration all technological and business requirements, such as the ones identified in the previous chapters of this document.

FEs are of the same TYPE if they consist of the same grouping of functions. The same FE Type may occur more than once in a model, and may also appear in the model of more than one service.

FEs are described in step 1 which consists of:

- a brief definition of the FE
- a description of what the FE has been put there for
- a global list of functions performed
- the list of relationship with the FE

Each interaction between a communicating pair of FEs in a model is termed an information flow. The RELATIONSHIP between any communicating pair of FEs in a model is the complete set of information flows between them.

FE Relationships are described in step 2.

Relationships are assigned TYPE identifiers which uniquely identify specific sets of information flows within the model. The same relationship type may occur more than once in a functional model.

If the model does not show a line between FE1 and FE2, there is no relationship between the 2 FEs. However FE1 and FE2 could still cooperate indirectly in the execution for a service if FE1 and FE2 are both communicating to a third FE.

If a communicating pair of FEs is located in physically separate devices, the RELATIONSHIP between them defines the information transfer requirements for a protocol between the devices.

FE ACTIONS are the elementary functions performed within an FE which are externally visible.

FE Actions are identified by unique FE Action Reference Numbers.

FE Actions are explicitly or implicitly notified to other FEs.

FE Actions are represented on the information flow diagrams (step 2) and SDL, diagrams (step 3) by brief prose statements. They are further refined for each FE Action Reference Number in step 4.

3.6.2 Distributed functional plane modelling

3.6.2.1 Overview of the IN functional architecture

The functional architecture of an IN-structured network contains the following FEs:

Call control related functions:

SSF - Service Switching Function:	This function interfaces with CCF and SCF. It allows CCF to be directed by the SCF.
CCF - Connection Control Function:	This function refers to call and connection handling in the classical sense (e.g., that of an exchange).
CCAF - Call Control Agent Function:	This function provides the user access to the network.
SRF - Specialised Resources Function:	This function provides a category of resources for access by users. Examples of resources include DTMF sending and receiving, protocol conversion, speech recognition, synthesised speech provision, …

Service control related functions:

SCF - Service Control Function: This function contains the IN service logic programs and handles service related processing activity.

SDF - Specialised Database Function: It provides the specialised data for realtime access by the SCF. It hides from the SCF the real data implementation and provides a logical data view to SCF.

Management related functions:

SCEF - Service Creation Environment Function: This function allows an intelligent network service to be defined, developed, tested and input to SMF. Output of this function involves service logic and service data template and service trigger information.

SMAF - Service Management Access Function: This function provides an interface to the SMF.

SMF - Service Management Function: This function involves service deployment, service provisioning, service control, billing and service monitoring.

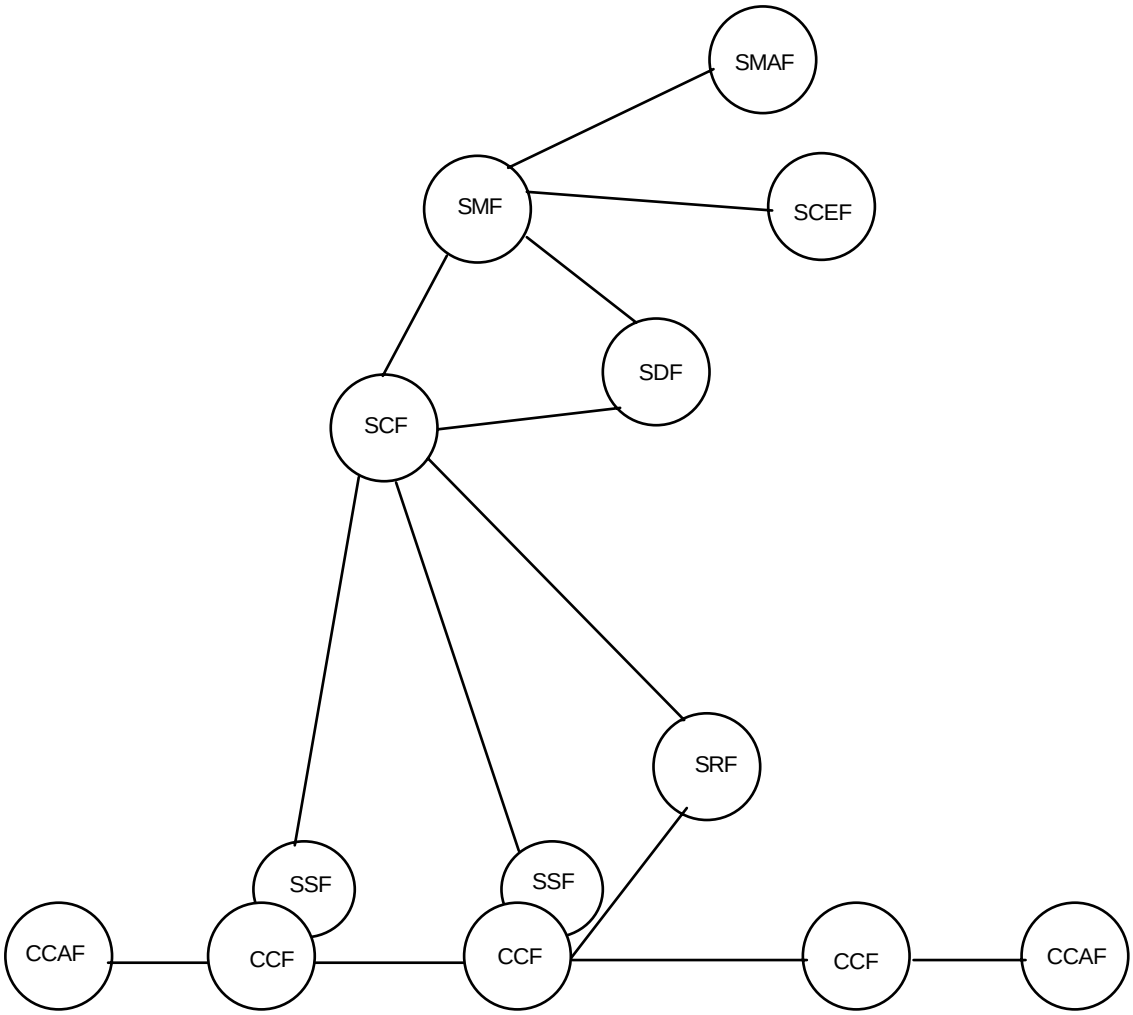


Figure 26: IN Functional Architecture

3.6.2.2 Description of FEs (step 1)

3.6.2.2.1 Service switching function / connection control function / call control access function

3.6.2.2.1.1 Definition

SSF: The service switching function (SSF) is the functional entity, embedded in a switch, which provides the switch connection-handling function with access to the service logic which can be located in the same switch or in a remote network element.

CCF: The CCF refers to call and connection handling in the classical sense (e.g. that of an exchange).

CCAF: The set of call control access functions (CCAF) provides the user access to the network.

3.6.2.2.1.2 Description

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.

3.6.2.2.1.3 Functions performed

The functions described in this subclause can be grouped into three parts:

- service switching functionality
- connection control functionality
- call control access functionality

Service switching functionality:

The functions in the SSF are related to the communication with the SCF. The SCF is offered a limited aperture of vision and influence into these functions. The following functions support interaction with the SCF:

Connection function

Requests from the SCF cause establishment of a connection between addressable entities in the network, e.g. an end-user, or an information source, etc. How the connection is established, is described in the connection control model in subclause 3.6.5.

Primitives that cause actions in the connection function are described in Annex C.

Event management and monitoring function:

If an event occurs in the SSF (e.g. reception of a particular signalling message, expired timer) it will be reported to the SCF. The event management and monitoring function supports the following capabilities:

- translation of signalling messages to events;

- generation of events (signalling messages);

- event mode handling (e.g. intercepted, duplicated, transparent, ignored).

Primitives that cause actions in the event management and monitoring function are described in subclause 3.6.5.

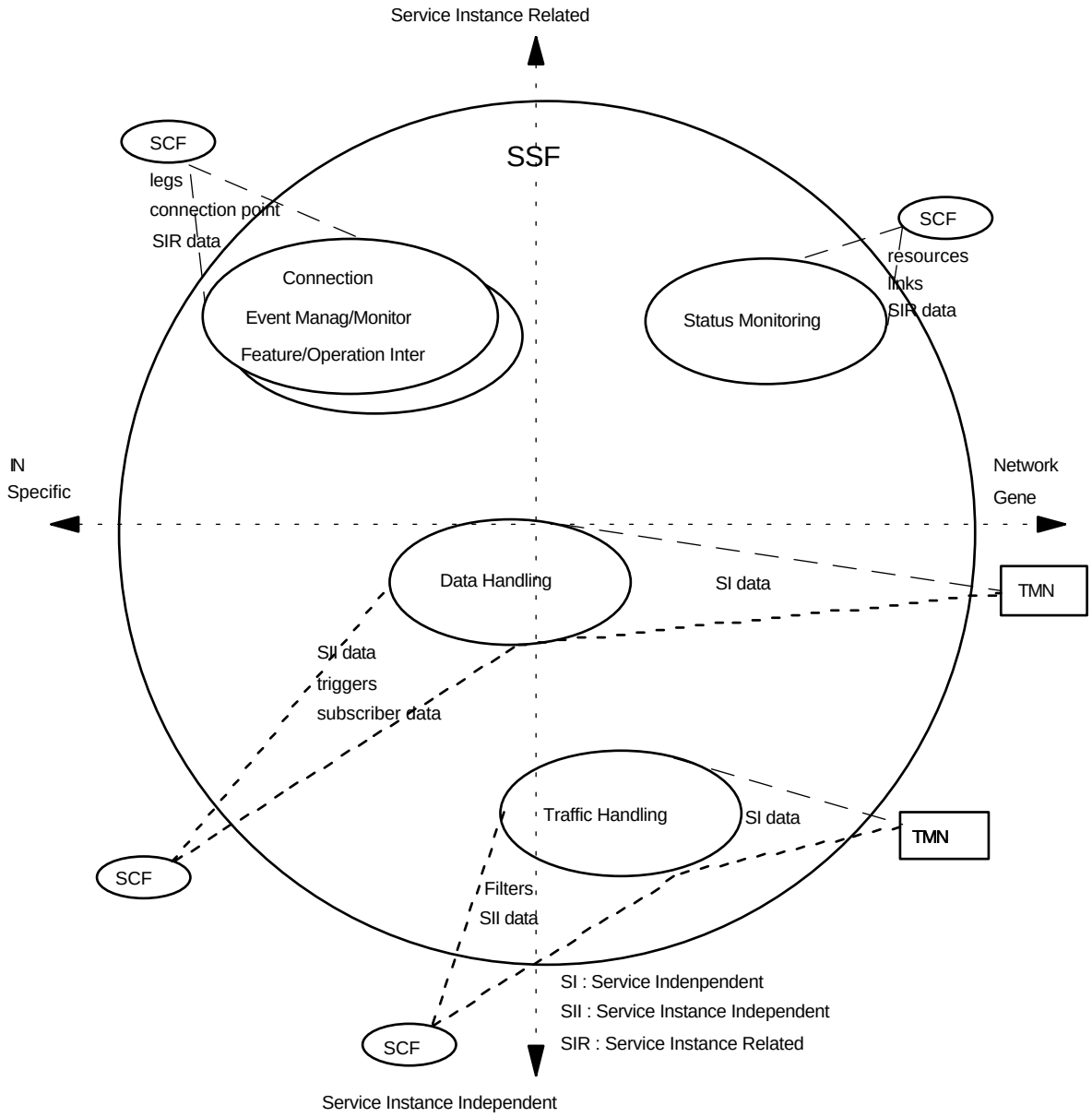


Figure 27: A view of the functions in the SSF and their relations to the SCF and the TMN

Status monitoring function

This function allows status monitoring of resources involved in call processing. It monitors queues, addressable entities and resources for communication with addressable entities (links). The status information will indicate whether a resource is busy, free or in reserved state.

Primitives that cause actions in the status monitoring function are described in Annex C, Clause 7.

Data handling function

The data handling function handles:

- service independent data (e.g. network management data)

- service instance independent data (e.g. trigger, charging and statistics data)

- service instance related data, i.e. specific call related data

Primitives that cause actions in the data handling and monitoring function are described in Annex C, Clause 6.

Traffic handling function

This function manages traffic limitation in an overload situation. It will contain activities such as overload decision, filtering of calls for certain service types, and filtering of calls for certain destination and origination.

Primitives that will cause actions in the traffic handling function are described in Annex C, Clause 6.

Feature and operation interaction function

This is the function that provides mechanisms to manage interactions between multiple IN features simultaneously active on a single call, and between multiple and non-IN features simultaneously active on a single call. A characteristic task of the feature and operation interaction function would be compatibility check and precedence check of the features involved in the service execution. It also handles interaction between IN features residing in different SCFs that are simultaneously active on a single call.

In figure 27 the functions in the SSF are shown together with the relationships to the SCF and the TMN. Data has been grouped into the two following categories:

- IN specific data, i.e. service instance related (SIR) data and service instance independent (SII) data

- Network generic data, i.e. service independent (SI) data, which concerns TMN.

Call Control Access Functionality

The set of network access functions provides entry points to IN for users. The following is a tentative list of functions:

Signalling detection function

- detection of signalling messages (e.g. analogue subscriber line messages, DSS1 messages).

Connection Control Functionality

The CCF describes the sequence of activities needed for the execution of a basic call. An important task is to identify standard trigger points in the basic call. The following is a tentative list of functions:

Basic Connection function

This function provides basic connection control to establish communication paths for users and interconnect such communication paths, e.g.

- manage digit collection according to numbering plan;

- analyse digits for destination;

- checking of compatibility (e.g. bearer service);

- select route from route list

- establish and maintain the connection

- provide busy indication and alerting indication

Trigger detection function

A trigger is an event which can cause IN features to be invoked. The trigger detection function determines that the conditions for an IN call are met. If so, the service request and related information is forwarded to the SCF. The trigger detection function may include screening of certain originating calls, etc.

Signalling interworking function

- DSS1 to ISUP, etc.

Synchronisation of resources

This function supports synchronisation of events between a user and the SRF (e.g. when the user is ready to receive an announcement).

Security function

- authentication

- ciphering

Service interaction function

- handling of reference information between CCFs

3.6.2.2.1.4 Relationships

SSF: The SSF has relationships with the SCF and the CCF. The relationship with the SCF will be subject for standardisation. When an IN-supported service is invoked, the SCF is offered a limited aperture of vision and influence through the following application contexts (for description see Annex C, Clause 6):
- connection control

- data management

- traffic management

- status monitoring

Since it is anticipated that the SSF and the CCF will be mapped into the same physical entity, the SSF-CCF relationship is not subject to standardisation.

Relationship between SMF and SSF is for further study. The SSF may have a TMN-Q3 interface.

CCF: The CCF has relationships with the SSF (see text above), SRF, CCAF and the CCF. The relationship with the SRF will be subject to standardisation.

CCAF: The CCAF has relationship with the CCF. The relationship is subject to standardisation (DSSI).

3.6.2.2.2 Specialised resource function

3.6.2.2.2.1 Definition

The Specialised Resource Function, SRF, is a functional entity (FE) which provides flexible specialised service independent resources for access by users via SSF and over transmission paths controlled by the CCF.

3.6.2.2.2.2 Description

The functions included in this FE mapped on to Service Independent building Blocks (SIB) as defined in the Service Building Block Plane.

The SRF groups the set of resources used:

- to receive information from,
- to send information to and
- to convert information from 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.

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.

Issues:

- the necessary synchronisation mechanisms between reception and sending i.e. the level of autonomy of the SRF is for further study.
- the provision of external programmability capabilities with the related programming interface is for further study.

3.6.2.2.2.3 Functions performed

Examples of SRF capabilities include:

- DTMF receivers,
- Customised announcements,
- Synthesised voice/speech recognition devices,
- Protocol converters,
- Interactive prompting facilities (oral or visual e.g. Videotex),
- Text-to-speech synthesis,
- Conference bridges.

In figure 28, the functions in the SRF are shown together with the relationships to other PEs.

3.6.2.2.2.4 Relationship

In IN-structured networks, the use of SRF resources is placed under the control of the Service Control Function (SCF).They may be invoked for several services and/or network users. Conversely several SRFs can participate in more complex service provision, but they are unaware of each other and thus not capable of communicating with each other.

FFS: the management of these resources may be placed under the control of the Service Management Function (SMF). This includes the management of customised announcement data and Videotex menus.

Thus, the SRF has relations with the following entities:- CCF: to exchange information with the CCF to set up the transmission path (bearer control) and to provide users with service support over the transmission path controlled by the CCF (service provision).

SCF: to exchange information with the SCF, to control the use of the service (service control). The SRF could be accessed by several SCFs.

SMF(FFS): to exchange management information (management control).

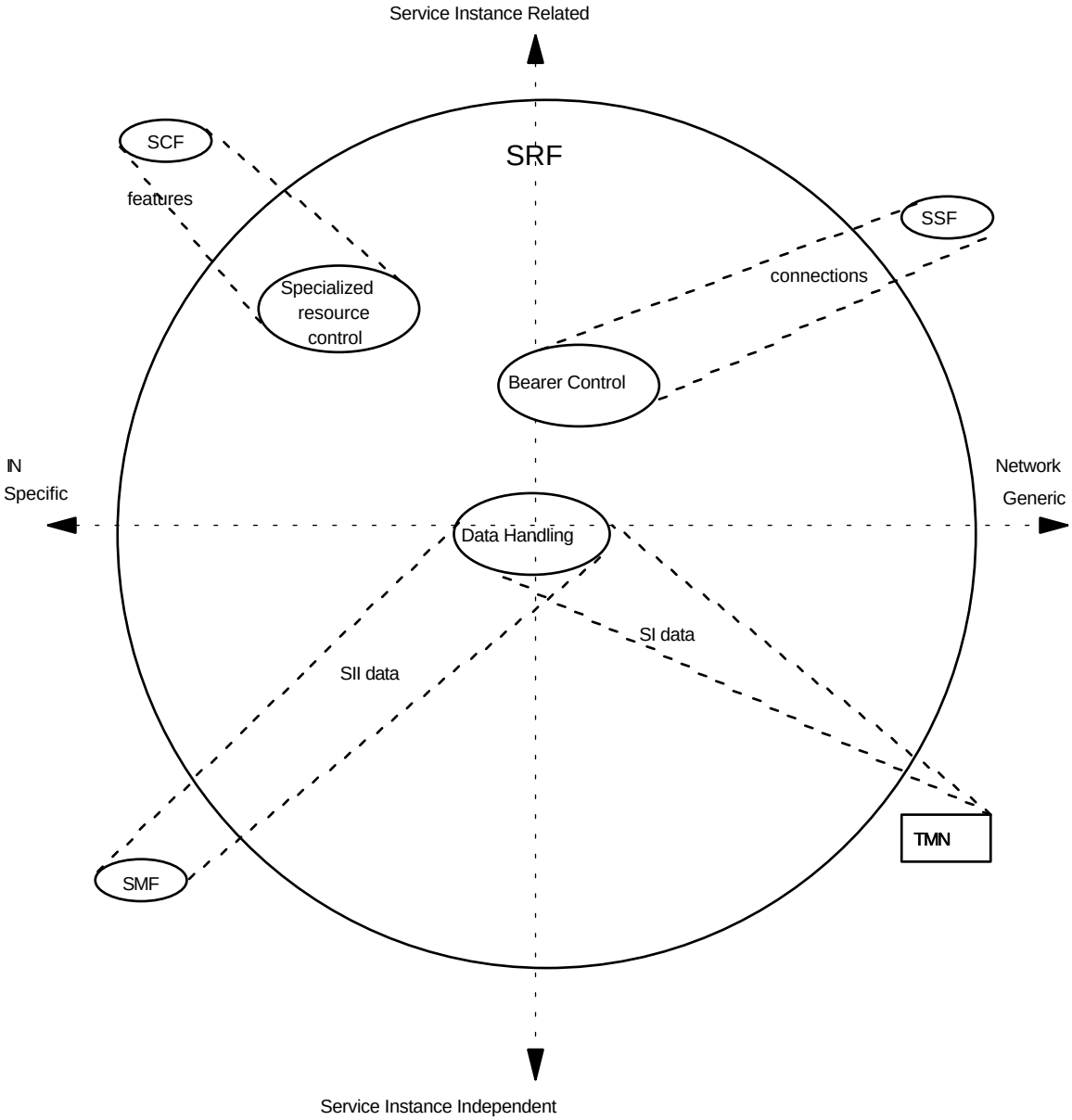
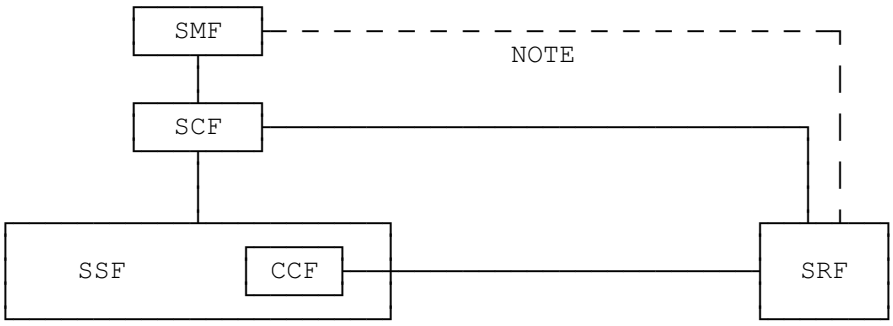


Figure 28



NOTE: Existence of this relationship is FFS

Figure 29

3.6.2.2.3 Service control function

3.6.2.2.3.1 Definition

The Service Control Function (SCF) provides for the execution of Service Logic. It contains the IN Service Logic Program (SLP) and handles service related processing activity.

3.6.2.2.3.2 Description

The Service Control Function (SCF) is a functional entity which provides the functionality for the control of IN service procession 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.

A functional model of SCF is given in figure 30.

3.6.2.2.3.3 Functional performance

SCF provides two types of functions, namely the Service Logic Execution Manager function, and the FE Access Manager.

1) Service Logic Execution Manager function

   The Service Logic Execution Manager (SLEM) function, as part of the Service Logic Execution Environment (SLEE), executes the service logic contained in a Service Logic Program in order to provide a specific service in an IN structured network. It is capable of handling several service logics, and for each service logic, several instances. These instances are SLP instances.

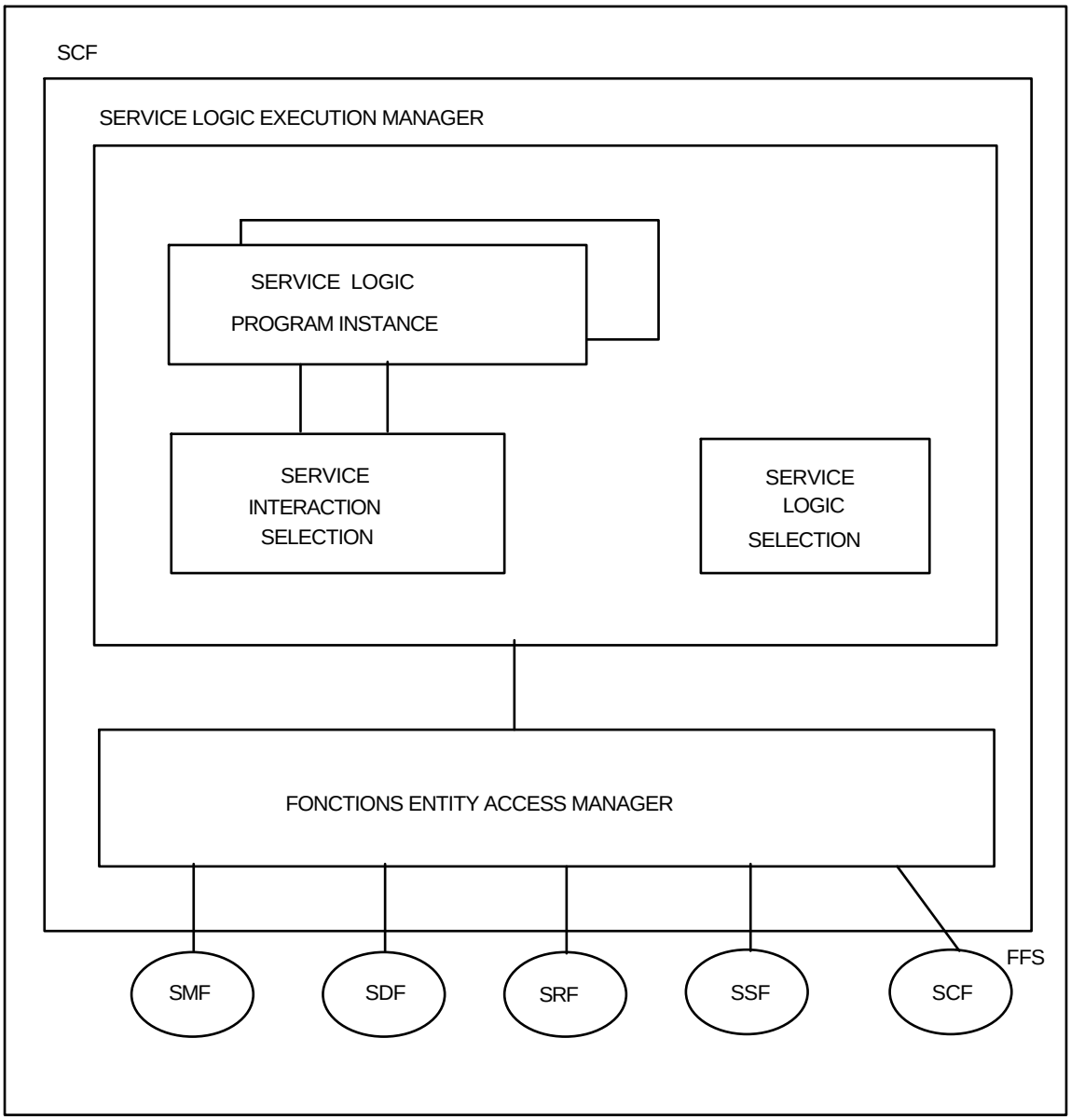


Figure 30: Functional Model of SCF

SLEM provides the functions for Service Logic (SL) interaction and selection:

- SL Interaction Manager function

  SL Interaction Manager (SLIM) is needed to manage and control the interactions arising from simultaneous execution to SLPs in the same SCF.

- SL Selection function

  SL Selection (SLS) is needed to relate a SLP for execution to realise a particular service. The selection of SLP may be determined by a service key.

2) FE Access Manager function

The FE Access Manager function provides bidirectional access to other FEs:

SMF: for the management of Service Logic Program (SLP) being executed, and for forwarding statistics or billing information.

SDF: access to specialised databases related to the SLP being executed.

SRF: access to specialised resources.

SSF: access to Service Switching Functionality, for invocation of SSF functions (connections, status monitoring, data handling, etc.).

3.2.2.3.4 Relationships

In order for SCF to provide the necessary functions for the execution of service logic, SCF has the following relationships with the other functional entities (FEs), namely Service Management Function (SMF), Specialised Database Function (SDF), Specialised Resource Function (SRF) and Service Switching Function (SSF).

The relationships are depicted in figure 31.

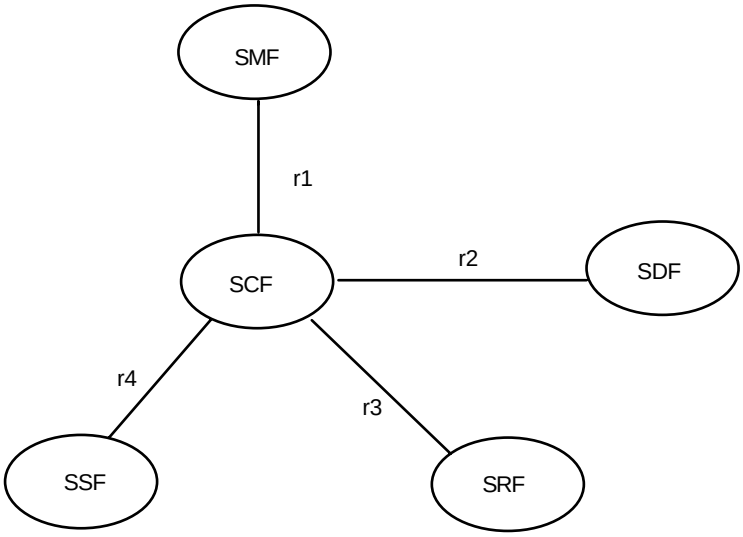


Figure 31: SCF Relationships with other FEs

r1 (SCF-SMF)

This relationship provides the interaction necessary for the management of Service Logic contained in SLP and related resources, as well as statistical and billing information. A functional model and description of SMF is given in subclause 3.6.2.2.7.

r2 (SCF-SDF)

This relationship provides the access to specialised database related to the execution of SLPs. A functional model and description of SDF is given in subclause 3.6.2.2.4.

r3 (SCF-SRF)

This relationship provides the access, control and use of specialised resources of SRF. A functional model and description of SRF is given in subclause 3.6.2.2.2.

r4 (SCS-SSF)

This relationship provides the interaction with SSF for the control of IN service processing, invocation of SSF functions, and for access to SSF capabilities and resources. A functional model and description of SSF is given in subclause 3.6.2.2.1.

3.6.2.2.4 Specialised database function (SDF)

3.6.2.2.4.1 Definition

The specialised Database Function provides the Specialised Data (Network and Customer related data) for real time access by the SCF. It hides to the SCF the real data implementation and provides a logical data view to the SCF.

3.6.2.2.4.2 Description

The SDF provides the capability for the rapid access to the information necessary to provide IN type services. The SDF may be located with the SCF in a SCP for instance, or located in a database that may be accessed by several SCPs.

Examples of SDF usage may be:

- IN Call number translations
- IN Call number "barred"
- Call diversion as in UPT
- "Congested trunks" and "time of day" routing data
- Directory enquiry information
- Short code dialling data
- Credit card authorisation

3.6.2.2.4.3 Functions performed

Management of physical database systems

The database may be "distributed" within the network.

For instance, the database system may be accessed from "centralised locations" (e.g. Network control) to change IN related data such as trunk translations or read traffic statistics. Other centralised access could be by the Service Control to introduce or change service parameters

Database update

The "Database" may be updated by different classes of authorised customers, and network operators with various access rights.

For instance, during the operation of certain services such as call diversion or UPT, it would be necessary to execute frequent Database updates to determine the final destination for a customer's incoming traffic. In such circumstances it may be efficient and advantageous to allow the service user, or customer, the facilities to directly modify the database via the SMF relevant routing information.

Security Considerations

The database must be adequately protected in a number of circumstances. A malicious or illegal access to the database could cause serious and costly disruption. Consideration should be given to guarding against the effects of inadvertent errors by authorised users. Different level of security may be associated with different types of customers data.

An example would be a user data entry for a Personal Identification Number (PIN), that must have restricted access for "read" or "write" authority.

Database failure protection

There should be a "hot" standby or back-up to the database available for immediate substitution in case of corruption or failure.

Types of data handled:

The function of the database is to provide access to the following types of data:

- Customer specific data

  These data may have an impact on the way service logic is executed.

- Service generic data.

- Some network data.

3.6.2.2.4.4 Relationships

The SDF has primarily a relationship with the SCF for data access.

The SDF has also a relationship with the SMF for the data update management.

3.6.2.2.5 Service creation environment function

3.6.2.2.5.1 Definition

The Service creation environment function allows an intelligent network service to be defined, developed, tested and put into SMF. Output of this function involves service scripts that is: services logic programs, service data template and service trigger information.

3.6.2.2.5.2 Description

The service creation environment function (SCEF) is the functional entity (FE), which allows an intelligent network service to be specified, developed, verified and input to the service management function (SMF).

3.6.2.2.5.3 Functions performed

Service specification function

The specification function provides the means to describe a service (with SIBs) on a high level and verifies the description (called "service logic") according to defined rules. Furthermore the function makes it possible to enhance service logic with SIBs and/or service logics to allow for further refinement in the service development.

Service Description

To ease service definition and modification, a graphical interface is suggested as environment into which the service is specified or modified.

Service Analysis

The service description is analysed with respect to logic, internal consistency and potential performance problems.

Specification Generation

A service specification is generated which consists of the following:

- service specification documentation
- service logic description
- service data description
- service trigger information (condition description or parameters)
- service test description

Service Development Function

The service development function takes as input the information provided by the service specification function. With those it provides the means to generate SLPs (combination of various use of SIBs with sequential and conditional statements) and service data templates.

Service Logic Programs Design

The service logic description and the service trigger information are taken as input to design the service logic programs, built with service independent building blocks, which are glued together by means of sequential and conditional programming.

The Service Logic Programs Design process generates a complete description of the service logic programs and a description of the service data parameters. SLPs are used by the SCF.

Service Data Template Design

The service data description is used to generate service data template. Service data template can be used by the SDF.

Service Verification Function

The service verification function takes both the output from the service specification function, and the output from the service development function as input for the verification process.

Service logic program verification

The service logic program processing in the network is simulated and the results are compared against the service specification. Any interactions with other services as specified by the test specifications are simulated and verified.

Service Data template verification

The service Data template processing is simulated to verify its correctness. The user interface, parameter checking and service data operations are verified in this process.

Service Creation Administration Function

This function provides administrative functions to administrate the service creation process. It provides library support for the input and output of the different functions. It keeps track of the state of the scripts

so only scripts which have passed the verification and are "approved" can be forwarded to the Service Management Function.

3.6.2.2.5.4 Relationships

The service creation environment function has a relationship with the Service Management Function (SMF), the service specified, the service designer and the service verifier. Only the relation SCEF-SMF is to be standardised.

3.6.2.2.6 Service Management Agent Function (SMAF)

3.6.2.2.6.1 Definition

This function provides an interface to the SMF.

3.6.2.2.6.2 Description

The SMAF may reside in an intelligent terminal or a computer. This physical entity would be connected to the physical entity where SMF resides. The purpose of the SMAF specification is to allow the interface of this connection to be specified.

3.6.2.2.6.3 Functions performed

Work Station Functions may be grouped into two areas, viz.

a) Human Interface Functions

   No detailed description of the human interface functions will be provided in order not to unnecessarily constrain vendor innovation.

b) Management Support Functions

   i) Human Interface Management

      - display manipulation (graphics, menus, textual representation ...)

      - user profiles

      - filter definition

      - definition of actions, schedules, ...

      - "help functions"

   ii) System Management

      data back-up

      user definition, authorisation

   iii) Service Management Control Supports

      All functions related to the usage of SMF capabilities. This includes:

      Service Management Control

      Service Provision Control

      Service Deployment Control ...

3.6.2.2.6.4 Relationships

The SMAF has a direct relationship only to the Service Management Function (SMF). By means of this relationship the operator can access and manipulate service data and service logic programs.

3.6.2.2.7 Service management function

3.6.2.2.7.1 Definition

The Service Management Function (SMF) is the functional entity (FE) which involves activities for service deployment, service provisioning, service control, billing and service monitoring.

3.6.2.2.7.2 Description

Service deployment activities are performed when introducing a service into the IN-structured network in a customer independent way.

Service provisioning activities are performed when a service is accommodated and provided to customers.

Service control activities are performed to ensure reliable and proper operation of a service.

Billing activities perform service independent billing operations like collecting, generating and modifying billing data.

Service monitoring activities perform service independent monitoring operations like observing, deleting, defining, validating and collecting service measurements.

The five distinguished activities of the SMF functional entity can be depicted in a schedule as shown in figure 32.

The final service creation activities in the SCEF are followed by the first service deployment activities within the SMF. The service deployment activities are completed with a set of testing activities which are also pointed at the remaining four activities. During these testing activities all activities required for a proper introduction and operation of the service within the SMF are tested. After a positive test result the service will be provided to the customers. A further explanation to figure 32 is given in the next subclause.

3.6.2.2.7.3 Functions performed

The service deployment functions

The basic task of service deployment is the introduction of service scripts and service generic data in the IN structured network in a customer independent way.

Within service deployment the following activities are distinguished:

- Service scripts allocation

  Allocation of service script as determined by the network operator.

- Service generic data allocation

  Allocation of service generic data as determined by the network operator.

- Signalling routing data introduction and allocation

For further study (the global title, in a STP, may have to be updated when some data are updated, or when one SCP is replaced by another. Two scenarios can be identified for updating STPs (possibly via STP Management). In the first scenario updates are performed via the network management: then the network management should be informed by the service management. In the second scenario updates are performed directly by the service management: the SMF must have a relationship with the STPs).

- Trigger date introduction and allocation

  Adjustment of trigger data (CCF/SSF) to the new situation in the IN structured network.

- Special Resource data introduction and allocation for further study.

  (Adjustment of special resources (SRF) to the new situation in the IN structured network in case the service to be introduced requires special capabilities during operation).

- Service test

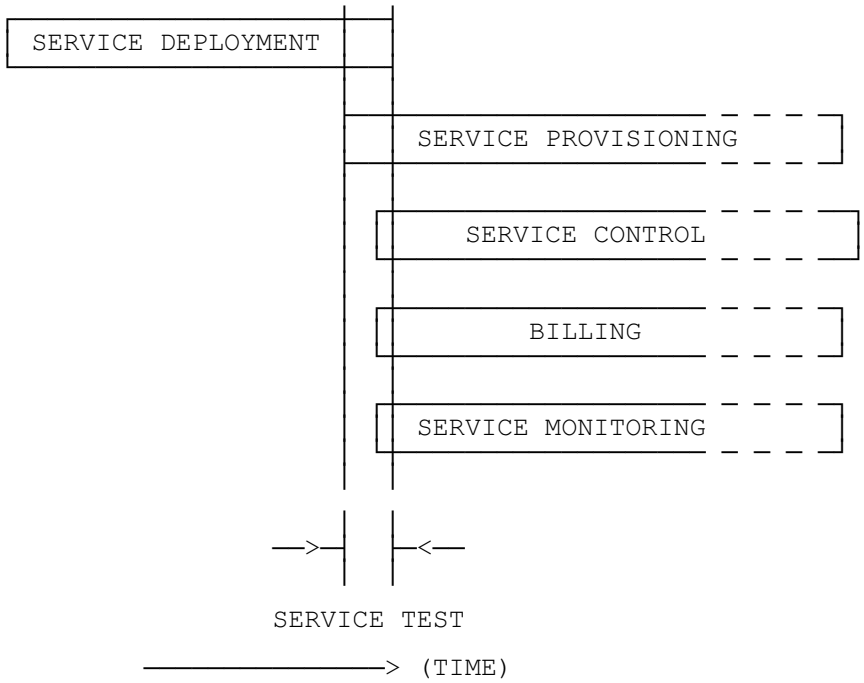


Figure 32: The activities of the SMF for the deployment, provisioning control, billing and monitoring of a service

The service test in the SMF is a continuation of the service test done in the SCEF. The service test in the SCEF was done in a test environment representing an IN structured network. The service test in the SMF will be done in the "real" IN structured network with possibly a representative number of test customers. During the service all SMF activities will be tested in coherence with both the service to be introduced and the services already provided in the IN structured network.

The service provisioning activities

After the customer independent installation of the service within service deployment, it has to be accommodated to the customer. This mainly means the introduction of customer specific data by the network operator or customer request.

Within service provisioning the following activities are distinguished:

- Customer specific data introduction and allocation

  During this activity the customer specific data is introduced and allocated (as determined by the network operator). The customer specific data refers to the information that is both related to and used by a customer (e.g. name, address, telephone number, bank account, and parameter values of services such as date, time and telephone numbers for the time-of-day-routing service).

Service control functions

A service, when provided to the customers, requires a number of activities in order to maintain proper operation.

The service control, billing and service monitoring activities start slightly later than the service provisioning activities (see figure 32), since they can not start before a service is provided to customers.

Within service control the following activities are distinguished:

Service maintenance

Service maintenance includes the following activities:

- Software maintenance

  Software maintenance consists of the modification of service logic (modification of service logic is a function of the SCEF). Introduction of modified service script in the IN structured network is done in service deployment.

- Updating service generic data

  The adjustment of service generic data by the network operator.

- Updating customer specific data

  The adjustment of the customer specific data by the network operator and/or customer. This depends on the kind of customer specific data. For instance, a change of address will be adjusted by the network operator, but a customer will be able to modify (on-line) the parameter values of the service.

- Updating signalling routing data

  For further study (the global title, in a STP, may have to be updated when some data are updated, or when one SCP is replaced by another. Two scenarios can be identified for updating STPs (possibly via STP management). In the first scenario via the network management: then the network management should be informed by the service management. In the second scenario directly by the service management: the SMF must have a relationship with the STPs.

- Updating trigger data

  The adjustment of trigger data (CCF/SSF) by the network operator. For instance, these adjustments may be necessary because of adjustments made in service generic data and customer specific data.

- Updating special resource data

  For further study (the adjustment of special resource data (SRF) by the network operator. For instance, these adjustments may be required as consequence of adjustments made in service generic data and customer specific data).

- Adjustment of the SMAF

  The customer/network operator interface to the SMF is provided by the Service Management Agent Function (SMAF).

  The interface to the customer and network operator has to be accommodated to the adjustments in their data. For instance, a customer who has changed of peripheral-type (customer specific data: DTMF-telephone to VTX-terminal). This change of peripheral also may cause a change of menu options.

- Service reconfiguration

  This activity consists of the re-allocation of service scripts, service generic data and customer specific data. For instance, the reason for service reconfiguration could be a change in the network configuration or improvement of the performance of services.

- Service (de)activation

  This activity gives the network operator the possibility to (de)activate (part of) a service temporarily. For instance, for maintenance purposes a televoting service which is only used on set times.

- Service dismantlement

  A service will be definitive taken out of operation when it becomes evident that the service does not satisfy any more, and can not or does not have to be adjusted. (This activity is reason for the different finishing times between service control and the other functions as shown in figure 35).

- Security

  In the SMF two types of security can be distinguished: access control and data control. Access control covers the identification, authentication and authorisation (command control) of both customer and network operator. Data control covers the control of the input of data by both the customer and the network operator.

Billing functions

The basic task of the billing function is to perform service independent billing operations like collecting, generating and modificating billing data.

Within billing the following activities are distinguished:

- Generating and storing charging records

  There are various operations that can be charged, for instance, the creation of a service (SCEF), the usage of a service (SSF/CCF, SCF), updating service generic data and/or customer specific data (SMF) and the generation of statistics (SMF). The charging records will be stored at the location they were generated.

- Collecting charging records

  The required charging records have to be obtained from the locations they are initially generated and stored. The collecting and processing of the charging records could be a part of the SMF. Another scenario consists of the collection of charging records by the SMF and processing by a separate Billing Centre Function which is outside the scope of IN.

- Modifying tariffs

  This gives the network operator the possibility of changing the tariff(s) of a service.

Service monitoring

During the operation of the service its functioning and usage has to be observed very censoriously. Service monitoring delivers information to both the network operator and the customer, but they have their own specific information requirements.

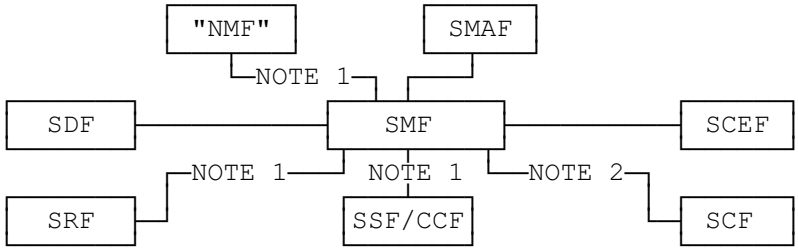
Within service monitoring the following activities are distinguished:

- Initiating measurements and collecting measurement data. The network operator and customer can initiate an observation of a service. A network operator needs in particular information to operate a service as efficient as possible (e.g. information about performance, faults, events). A customer needs in particular only information about the usage of the service (e.g. information about the availability of the service, the number of calls). The information needed can be specified via the SMAF (possibly via a menu).

- Analysis and reporting of measurement data. The collected measurement data have to be presented to the customer/network operator in a usable form (SMAF). This means an analysis of the rough measurement data to generate statistics (reports).

3.6.2.2.7.4 Relationships

The relationships of interest are depicted in a figure and are specified below.



NOTE 1: existence and relationships for further study

NOTE 2: content (information flows is ffs)

Figure 33: The relationships of interest of the SMF

SSF/CCF: The relationship between SMF and SSF/CCF is for further study. (Depends on the functionality of the SCF. In order to avoid a direct relationship between the SMF and SSF/CCF, the SCF may have to perform some transit functions between SMF and SSF/CCF. It is for further study whether this extra (management) functionality for the SCF is desired or not)

- updating trigger data

SCF: The relationship between SMF and SCF identifies e.g.:

- allocation of service logic programs;

- service (de)activation;

- collecting of billing records;

- initiating and collecting of measurement data;

- (it is for further study if transit functions in the SCF are desired, see also "relationship between SMF and SSF/CCF').

SCEF: The relationship between SMF and SCEF identifies e.g.:

- software (service logic/data) exchange;

- documentation exchange (manual).

SMAF: The relationship between SMF and SMAF identifies e.g.:- access to SMF (customer and network operator).

SDF: The relationship between SMF and SDF identifies e.g.:

- updating of service generic data and customer specific data.

SRF: The relationship between SMF and SRF is for further study.- (probably a relationship exists to update SRF specific data, but this may also occur via the SCF).

NMF: The relationship between SMF and Network Management Function (NMF, not an official FE of the DFP) is for further study. This relationship identifies the exchange of information between the SMF and network management (e.g. information about configuration, faults, traffic, performance, security). It is for further study if TMN is relevant to IN and SMF.

3.6.3 Overall call modelling: Introduction

Within the IN, a need has been identified for at least two "overall" call models. One will be used for service specification, and the other will be used for service execution. This subclause identifies the initial requirements for each of these models.

Another model may be required for service management aspects.

3.6.3.1 Requirements on the model to be used for service specification

One of the objectives of the IN architecture is to allow the specification of IN services in a way which ignores the actual configuration of the network.

Service Logic at the Global Functional Plane, uses SIBs, which are realised within the operational network, and provide a global view of network resources which is service, network implementation and vendor independent. One category of SIBs handles the establishment, release and control of calls.

A call model is therefore required which represents an abstract view of those logical resources involved in call processing which need to be known by the IN service specification/design function. This model does not constrain the implementation of call processing functions performed within the switches or other network elements.

This subclause defines the requirements for this call model.

a) Separation of concerns

The model should define only those resources (or objects) which participate in the processing of a call and may be observed and manipulated across the SIB boundary (interface A in subclause 3.1).

Objects and/or information which relate to call processing but have no meaning to the IN service designer/specifier, such as internal switch signalling events, should not be described in this model.

b) Ease of use

The service specifier is generally a programmer with little knowledge of the techniques and internal mechanisms of the network elements, and may have some difficulty in comprehending the jargon used by telecommunication experts to describe the processing of calls.

Thus the model should provide an easy to understand definition and way to manipulate "calls". The "call" object should be represented in a limited number of states and uniquely identified at the service specifier/designer level.

c) Transparency to switch type

The model should support all types of switches which could be present in an IN structured network. It should not make any assumption about how call processing is actually implemented in a switch.

Call "events" observable by the service designer should represent the functions and not the implementation mechanism.

d) Transparency to device type

The model should mask as much as possible from the service specifier the specifics of the network access points as these are irrelevant to call processing.

One technique is to define "Logical Devices" of only a few types, such as Directory numbers, trunks, ACD and Hunt groups.

e) Transparency to network type and topology

The model should support a logical view of "connections" to allow the execution of the same service over networks of different types and different topologies.

The ISDN access states (CCITT Recommendation Q.931) are felt appropriate as a basis for describing "connection" states and events.

f) Compatibility with other models

This model should take into account, and be compatible with, all the models currently being identified (e.g. connection control model).

3.6.3.2 Requirements on the model to be used for service execution

In order to prevent unwanted interactions between concurrently executing services, a call model will be required in the Distributed Functional Plane which defines the allowed relationships between services.

This subclause describes the requirements on this model.

a) Location independence

The model should be independent of the location of the SCF in a physical network element. The SCF could for example, be resident in the switch, or locally attached, or remote from the switch.

b) Support for different service execution scenarios

The model should support at least three possible scenarios for service execution.

i) Calls where each invoked IN service is controlled by a separate (independent) SCF.

ii) Calls where one SCF controls more than one invoked IN service in the same SSF.

iii) Calls where invoked IN service requires co-operation between several SCFs.

The above must also include the possibility that the functional entities involved in service execution do not reside in the same network.

These scenarios and others are described in more detail in the Annex D, Connection Control Modelling.

c) Compatibility with other models

This model should take into account, and be compatible with, all the models currently being identified (e.g. connection control model).

3.6.4 Overall call modelling scenarios

See Annex B (informative).

3.6.5 Connection Control Model (CCM)

3.6.5.1 General

The CCM provides a high-level service and vendor/implementation independent abstraction of IN service/feature processing in the Service Switching Function (SSF) and Call Control Function (CCF). This abstraction provides an observable view of SSF/CCF activities and resources to the Service Control Function (SCF), through which the SCF can interact with the SSF in the course of executing service logic.

3.6.5.2 CCM Objectives/Criteria

Some of the general CCM objectives/criteria are given below. The CCM should:

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

   - a generic 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;

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 actions within a service switching function, and

g) Provide rules for 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.

3.6.5.3 General assumptions

Scope of CCM

The scope of CCM will focus on SSF/CCF call/service processing functions, as well as other functions that may be required to support an IN call (e.g. Specialised resource function (SRF) functions).

Relation to IN conceptual model

CCM addresses functional entities (FEs) and their relationships in the Distributed Functional Plane. The relation to other planes will be better understood as the IN Conceptual Model definition progresses.

Use of CCM

The CCM is a tool used by IN architects to model a call and to understand and describe the distribution of functions between FEs and FE relationships. The observable call/service processing functions describes by modelling efforts can be used by service designers to facilitate the creation of service logic. These functions are reusable, in that the same function can be reused for a variety of services. In addition, given that a robust feature interaction mechanism is described, service designers would not be constrained by previously defined features, and would also be able to reuse these features to create new ones.

3.6.5.4 Proposed model

Annex C contains a detailed description of a CCM which complies with the objectives and criteria given in subclause 3.6.5.2. It is with noting that, even with another model, the same SSF-SCF interaction could have been described. The full description of the proposed CCM is contained in one location for consistency and completeness. Although it is considered to be stable, it is contained in an Annex as some parts of it are considered to be too detailed to fall within the scope of the main body of this Technical Report.

The material contained in the Annex forms a good basis for designing the SCF-SSF protocol. This is currently being studied by SPS3.

3.6.6 Examples of Call Configuration

See Annex D (informative).

4 I.N. terminology

4.1 Principles

Currently, a large amount of parallel activities are in progress on the subject of Intelligent Networks. These activities are not only proceeding in ETSI's STCs, but also in CCITT both Study Group XI and XVIII, as well as in the U.S. (e.g. Bellcore's Multi Vendor Interactions Forum and T1S1). Consequently, unique and widely accepted definitions for terms are essential to avoid misunderstandings in debates and other exchanges of information. To this end, NA6 proposes to produce definitions only for new terms essential to the successful progress of its studies and to accept already existing Terminology adopted by international organisations like CCITT and ISO if they are appropriate. NA6 will strive to achieve alignment with CCITT by requesting its members to submit contributions to the appropriate Study Groups of CCITT when it is detected that CCITT definitions do not exist or differ from the NA6 definitions.

The rapporteur for terminology will maintain a list of definitions which will be reviewed at every NA6 meeting. Terms will be accepted into the NA6 list of definitions when their use is established in NA6 Temporary Documents, Reports, ETSs, etc. It is not the intention simply to repeat terms used in the general literature on IN.

As definitions will be obtained from a variety of sources, they will be checked for consistency of use throughout the list (i.e. use of any defined word will be in the same context in any other definition in the list). Use of terms in the defined sense and in the vernacular sense in different parts of the terminology list will be avoided.

Definitions are being kept brief, but sufficiently detailed to avoid ambiguities.

A procedure is also established for deletion of terms which have become obsolete or the use of which is not recommended. This procedure is summarised by the IN Terminology Life Cycle diagram presented in figure D.5 of Annex D, based on a three CATEGORY system.

[1] Stable definition

[2] Definition needs refinement or usefulness of this terms is in doubt

[3] Term is not required for future work and is a candidate for removal from the list

The intention of this transition diagram is to stimulate additions, clarifications and removals in order to keep the list of IN terminology definitions up-to-date.

The transition will happen on a per term basis.

The list of IN Terminology definitions will also contain a reference to the source of each individual definition.

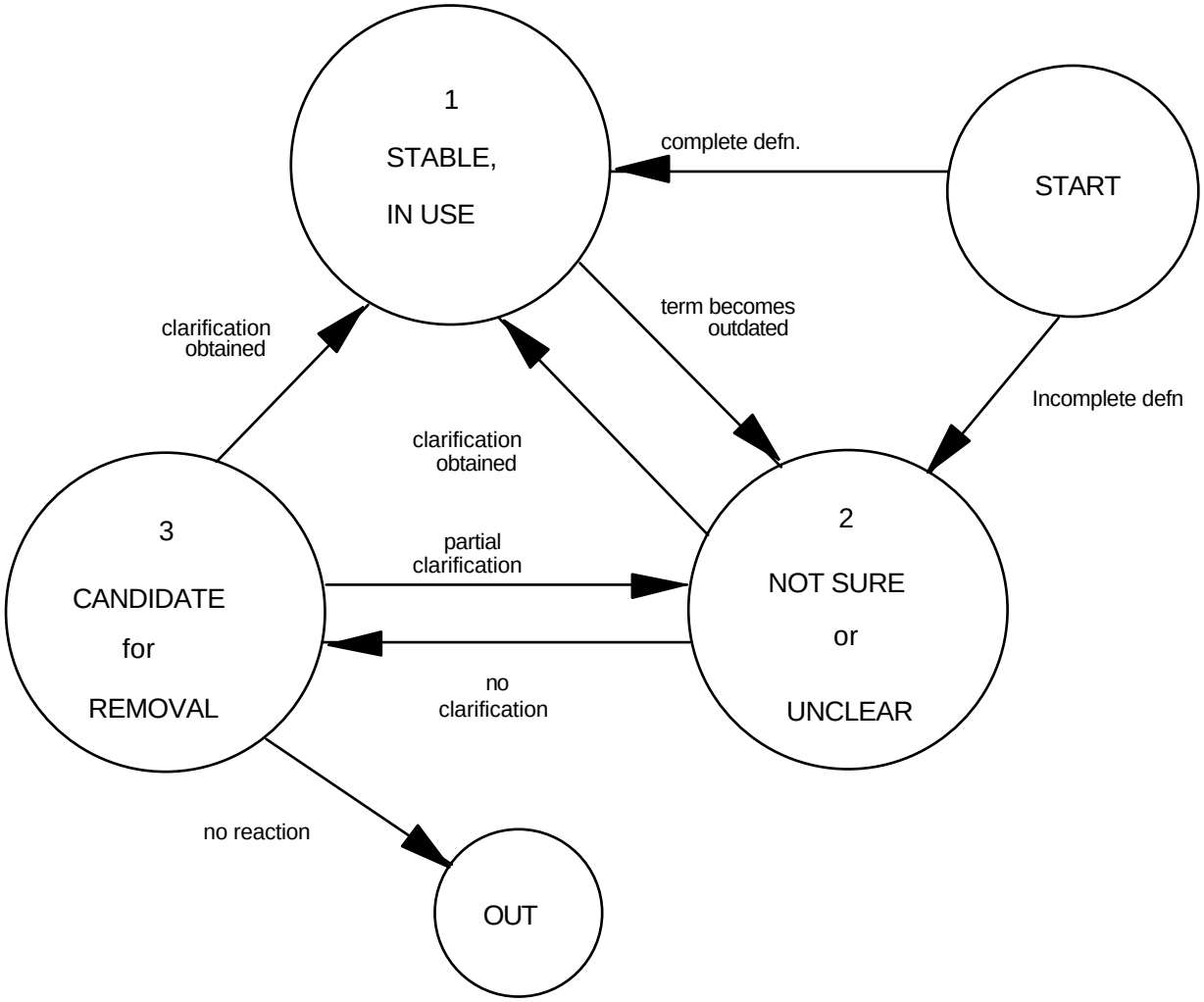

1
STABLE,
IN USE
START
complete defn.
term becomes
outdated
clarification
obtained
Incomplete defn
clarification
obtained
3
CANDIDATE
for
REMOVAL
partial
clarification
2
NOT SURE
or
UNCLEAR
no
clarification
no reaction
OUT
TERMINOLOGY LIFECYCLE

Figure 34

4.2 IN terminology glossary

The IN terminology glossary is found in table 4.

Table 4 (sheet 1 of 6)

Term	Abbr	Definition	Source	Category
Application		An entity in a network that provides a user with a set of capabilities that have an implementation - independent specification, to control the communication processing, and storage of information and to control a user interface.	Bellcore MVIF	2
Application Contexts	AC	The application context is a set of rules that govern communications requirements to allow the associations to be formed to provide the service requested. It is used to state just what functions are needed for a particular instance of communication.	CCITT Q11 Recommendation Q.11 XI IC 6225	2
Application Entity	AE	The aspects of an application-process pertinent to OSI.	CCITT Recommendation X.200 BB	1
Application Programme Interface	API	A software interface which is vendor, technology, network and service independent and is used to access service control functions for service introduction and/or removal.	3-TD-19	2
Application Service Element	ASE	A part of an application-entity which provides an OSI environment capability, using underlying services where appropriate.	CCITT Recommendation X.200 BBG	1
Association		A logical relationship between entities to perform a function.	CCITT	1
Associator Segment		An Associator Segment is one type of Basic Call Segment within the Socket Relationship Model. It represents the association between the end-users of the network and the basic all processing functionality concerned with forming a connection between two end-users.	NA6-WG4	1
Basic Call		A Basic Call is a call between two users that does not require features (e.g. a POTS call).	NA6-WG4	1
Basic Call Model	BCM	The Basic Call Model describes the sequence of activities needed for the execution of a basic call. The BCM is art of the Connection Control Model.	NA6-WG4	1
Basic Call Segment		A Basic Call Segment is one type of connection segment in the Socket Relationship Model which reflects the functionality of part of a basic call. There are two types, Terminator and Associator segments.	NA6-WG4	1
Call		The use, or possible use, of one or more connections set up between two or more users and/or services.	CCITT Recommendation SG IX	1
Call Control		Call Control is the coordinated activities of services control and connection control invoked in a call (e.g. provide service features and establish, supervise, maintain and release connections).	CCITT Recommendation SG XI	1
Call Control Agent Function	CCAF	The Call Control Agent Function provides the user access to the network.	NA6-WG3	2

Table 4 (sheet 2 of 6)

Term	Abbr	Definition	Source	Category
Call Control Function	CCF	The Call Control Function refers to call and connection handling in the classical sense (e.g. that of an exchange).	NA6-WG3	1
Call Model		The call model is an abstract model of logical states within which a generalised service is required to operate at as high a level of abstraction as possible. These states are compatible with the actual physical resources state.	2-TD-30 2-TD-65	2
Capability	CS	A Capability Set is defined as a set of IN capabilities that is to be the subject of standardisation activities and for which the availability of standards recommendations will be targeted for a particular time frame. The Long Term Capability Set (LTCS) is the CS for the Target IN Architecture.	NA6-WG1	1
Chain		A Chain is a component of the Socket Relationship Model. It consists of a number of connection segments joined serially by means of links. The SRM may comprise a single chain or a configuration of inter-connecting chains.	NA6-WG4	1
Connection Control	CC	Connection Control is responsible for setting up, maintaining and releasing the communication path between two or more users. A user may be a network entity (e.g. DTMF receiver).	2-TD-12	1
Connection Control Model	CCM	The Connection Control Model is a model that describes the static and dynamic aspects of the interaction between the SSF and the SCF in order to co-ordinate the call processing activities of the SSF in the support of IN services. The Connection Control Model is a unified model comprising three components: a) The Basic Call Model b) Socket Model c) Socket Relationship Model	NA6-WG4	1
Connection Control Socket Model		The Connection Control Socket Model is part of the Connection Control Model and models the Interaction which takes place between the SCF and the SSF within the bounds of a Connection Control Socket type in support of a feature operation.	NA6-WG4 1	
Connection Point		A Connection Point is an object accessible via a socket representing a logical bridge indicating which legs can logically communicate with each other, from the perspective only of the single feature acting through the socket. NOTE: A second service acting via another socket on the same call may have split the logical communications path, causing the end-to-end end path to be broken, but this is not known by the first service. A socket may contain more than one connection point. A connection point object may be created at the time of socket creation in order to represent an existing connection between two legs, or it may be created by the subsequent operation of the service logic. All connection points within a socket have a unique local identity.	NA6-WG4	1

Table 4 (sheet 3 of 6)

Term	Abbr	Definition	Source	Category
Connection Segment		The Socket Relationship Model is composed of Connection Segments, which can be chained together serially in various configurations. Each connection segment contains its own private virtual switch represented by leg and connection point objects. There are two types of Connection Segment: basic call segments and feature segments.	NA6-WG4 1	
Customer		A Customer is an entity which receives services offered by a service provider based on a contractual relationship. It may include the role of a network user.	NA6-WG5 2	
Data Template Programme	DTPA	Data Template Programme consists of service-data related logic, user data related logic, database related logic, etc. DTP is an interactive programme which interacts with the Network Operator or the customer and updates the data located at the Network Information Database.	3-TD-21	1
Distributed Functional Plane		The Distributed Functional Plane is one on the planes in the IN Conceptual Model which contains functional entities and their relationships.	3-TD-32	1
Elementary Function		A primary of basic function that cannot be further decomposed. One of the set of functions comprising global function.	CCITT Recommendation I.310	1
Feature		A Feature is a re-usable unit of one or more service capabilities forming all or part of a service (IN or switch based). With respect to the IN connection control model a feature is controlled by the SCF via a single socket instance.	NA6-WG4	2
Feature Interaction		A situation that occurs when an action of one affects an action or capability of another. Programmes and handles service related processing activity.	CCITT	1
Service Control Point	SCP	An entity in the Intelligent Network that implements a service control function.	CCITT	1
Service Creation		Service Creation is an activity whereby supplementary services are brought into being through the specification phase, the development phase and the verification phase.	NA6-WG2	1
Service Creation Environment Function	SCEF	The set of functions that support the service creation process, the output of which includes both service logic programs and service data templates.	CCITT	1
Service Data Function	SDF	The Service Data Function handles access to service related data and network data and provides consistency checks on data. It hides to the SCF the real data implementation and provides a logical data view to SCF.	NA6-WG3	2
Service Description		Service Description is the step which matures the definition of a service concept into a detailed service description from the user's perspective.	NA6-WG2	1

Table 4 (sheet 4 of 6)

Term	Abbr	Definition	Source	Category
Service Independent Building Block	SIB	The service independent building blocks are the result of the decomposition of services into service independent entities. These building blocks provide the flexibility to construct a variety of services.	NA6-WG5	
Service Logic	SL	A sequence of processes/functions used to provide a specific service.	CCITT	1
Service Logic Execution Environment	SLEE	The composite set of capabilities developed by a vendor which provides a functionally-consistent interface to Service Logic Programmes, independent of the vendor's underlying platform. It defines the run time environment on which SLP's are loaded and executed.	Bellcore MVIF	1
Service Logic Programme	SLP	A software programme containing Service Logic.	CCITT	1
Service Management Agent Function	SMAF	This is an alternative term for Work Station Function. The SMAF may reside in an intelligent terminal or a computer. The physical entity would be connected to the physical entity where SMF resides. The purpose of SMAF is to allow the interface of this connection to be specified.	NA6-WG3	1
Service Management Function	SMF	Service Management Function is the functional entity which involves activities for service deployment, service provisioning, service control, billing and service monitoring.	NA6-WG3	1
Service Operator or Provider	SO	The Service Operator also known as Service Provider has overall responsibility on service operation and on data base management (i.e. he commercially manages the service).	NA6-WG5	2
Service Plane		The Service Plane is part of the IN Conceptual model and contains the services as they are perceived by the customers.	NA6-WG5	1
Service Provider		The organisation that commercially manages the service and that interfaces with Service Customers.	NA6-WG5	1
Service Subscriber		This Service Subscriber subscribes to a service and is registered in the service database. He may have responsibility on some data value definition. NOTE: The SS does not need to be User of the Service he subscribes for (e.g. a Credit Card organisation that wants its members to be able to use Credit Card Calling).	NA6-WG5	5
Service Switching Function	SSF	The Service Switching Function is the function, embedded in a switch, which provides the switch connection-handling function with access to the service logic which can be located in the same switch or in a remote network element. This function interfaces with CCF and SCF. It allows CCF to be directed by the SCF.	NA6-WG3	2
Service Switching Point	SSP	A physical entity that implements a service function.	CCITT	1

Table 4 (sheet 5 of 6)

Term	Abbr	Definition	Source	Category
Socket		A socket is an instance of a logical point of access into the Service Switching Function through which service logic pertaining to a single feature invocation can operate. (The service logic itself forms the plug).	NA6-WG5	1
Socket Relationship Model		The Socket Relationship Model is part of the Connection Model and describes the relationships between the basic call processing functionality and multiple features acting simultaneously on a call. Each IN based feature is controlled by the SCF via a separate instance of a connection control socket type.	NA6-WG4	1
Socket Sub-type		The concept of socket sub-types allows constraints to be imposed on a socket type with regard to the numbers of logical objects (e.g. legs or connection points) which are accessible via a socket instance, and to the operations which can be performed on them.	NA6-WG4	1
Socket Type		A Socket Type is a particular class of socket with a specific set of properties. Socket types are differentiated from each other according to the types of objects accessible and the operations which may be performed on those objects. Every socket is an instance of a particular socket type. The socket type of prime importance is the Connection Control socket type in which the objects accessible are legs and connection points.	NA6-WG4	1
Specialised Database Function	SDF	The Specialised Database Function provides the specialised data for real time access by the Service Control Function. It hides to the SCF the real data implementation and provides a logical data view to the SCF.	NA6-WG3	1
Specialised Resources Function	SRF	The Specialised Resources Function provides a category of resources for access by users. Examples of resources include DTMF sending and receiving, protocol conversion, speech recognition, synthesised speech provision, ...	NA6-WG3	1
Switching		The process of interconnecting functional units, transmission channels or telecommunication circuits for as long as is required to convey signals.	CCITT Recommendation I.112 RB	1
Terminal		An equipment, located in close proximity to the user which presents to the user the information received from the network in a form compatible with the user's requirements and performs also the complementary function from the user to the network.	CCITT Recommendation I.112 RB	1
Terminator Segment		A Terminator Segment is one type of Basic Call Segment within the Socket Relationship Model. It represents the external access point of a logical transport path into a switching node (e.g. a subscriber line or trunk circuit) and the basic call processing functionality associated with dealing with originating and terminating calls from/to this access point.	NA6-WG4	1

Table 4 (sheet 6 of 6)

Term	Abbr	Definition	Source	Category
Transparency of Location		Transparency of location is a characteristic of a Network Architecture, which enables an implementation of a Functional Entity to interact with implementation of other Functional Entities without having knowledge of the actual physical location (local or remote) of the available implementations of these Functional Entities.	2-TD-15	1
Trigger		A stimulus for initiating an action.	CCITT	1
User		An entity external to the network that uses its service(s).	CCITT	1
Work Station Function	WSF	The Work Station Function provides for the human interface and the communication to the Systems Management Function (e.g. screen presentation, ..).	NA6-WG3	2

Annex A (informative): Workplan

A.1 IN work plan

This clause describes the workplan for the development of the technical specifications.

A.1.1 Definition of capability sets

A Capability Set (CS) is defined as a set of IN capabilities that is to be the subject of standardisation activities and for which the availability of standards will be targeted for a particular time frame. The Long Term Capability Set (LTCS) is the CS for the Target IN Architecture.

A.1.2 Scope and objectives

Three formal objectives stated for Intelligent Networks are as follows:

- vendor and technology independence;
- network implementation independence;
- service implementation independence.

However, these objectives must be reconciled with the statement that the specification of the initial phase of IN should consider the transition from current architectures (subclause 1.3 of main document).The purpose of this work plan is therefore to identify specific phases for the evolution of IN capabilities, leading to targets for the development of appropriate standards ETSs. Each phase should be a step in the smooth evolution towards a long term IN architecture.

A.1.3 General considerations on the standardisation process

In order to be able to project standards for the IN it is first necessary to understand how IN as a new technological concept can be introduced without disrupting the existing technology base. Figure A.1 shows a general diagram of the standardisation process, making a strong distinction between on the one hand the concepts and modelling and on the other hand the application of concepts and modelling to allow a transition from the current network towards a target IN.

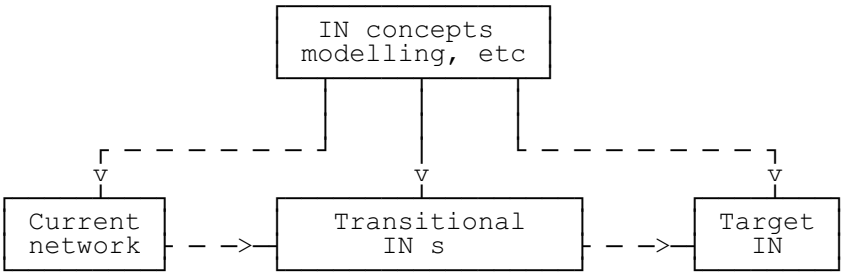


Figure A.1: Standardisation process

It is important to notice the following different inputs to the standardisation process:

- inputs from IN concepts & modelling (straight arrows):

  These are the network design principles that are necessary to meet the long-term objectives of the IN. By their nature these principles are likely to be stable in time, thought it is acknowledged that the long term objectives of the IN will need to be enhanced to encompass new technologies.

- input from the "previous" network in the transition path (dashed arrows):

  These are constraints that are necessary to build upon and evolve from existing ETSs towards ETSs for a target IN in a stepwise manner.

There will be two work efforts to implement the above standardisation process:

- A work effort to develop the architectural concepts and modelling tools, and

- A work effort to produce ETSs for each phase.

A.1.4 Standards areas

A standards outline can be derived from the above-described standardisation process. Three areas are identified impacting the production of ETSs:

Area 1: IN architectural concepts and modelling (IN Framework Document):

This area contains the results from the development of the IN target architecture, IN concepts, modelling techniques and other network design tools.

Area 2: IN transition planning and phase definitions:

Here a strategy is worked out (not necessarily resulting in a ETS) allowing a proper transition from the existing technology base towards a target IN infrastructure. As it will take many years to specify and develop ETSs for networks that meet all objectives of the IN (= "target architecture"), it may be appropriate to identify multiple standardisation phases. For each phase it is necessary to determine the service functionality (e.g. yes/no include service creation capabilities for the customer) and technological constraints (e.g. yes/no include ISDN packet switching in the D-channel), resulting in the definition of a CS.

Area 3: IN architecture and interfaces:

In this area the specifications are provided that are necessary for the implementation of IN equipment, interfaces, interworking, etc. For each phase an evolving set of ETSs may emerge.

The relationships between the areas are shown in figure A.2.

It should be observed that for each standardisation phase the information flows, interface specifications and network element specifications are to be developed on the basis of the service plane architecture and the global functional plan architecture as agreed to apply for that particular phase. The distributed functional architecture and call model specified at a high level in area 1, will need to be detailed in area 3.

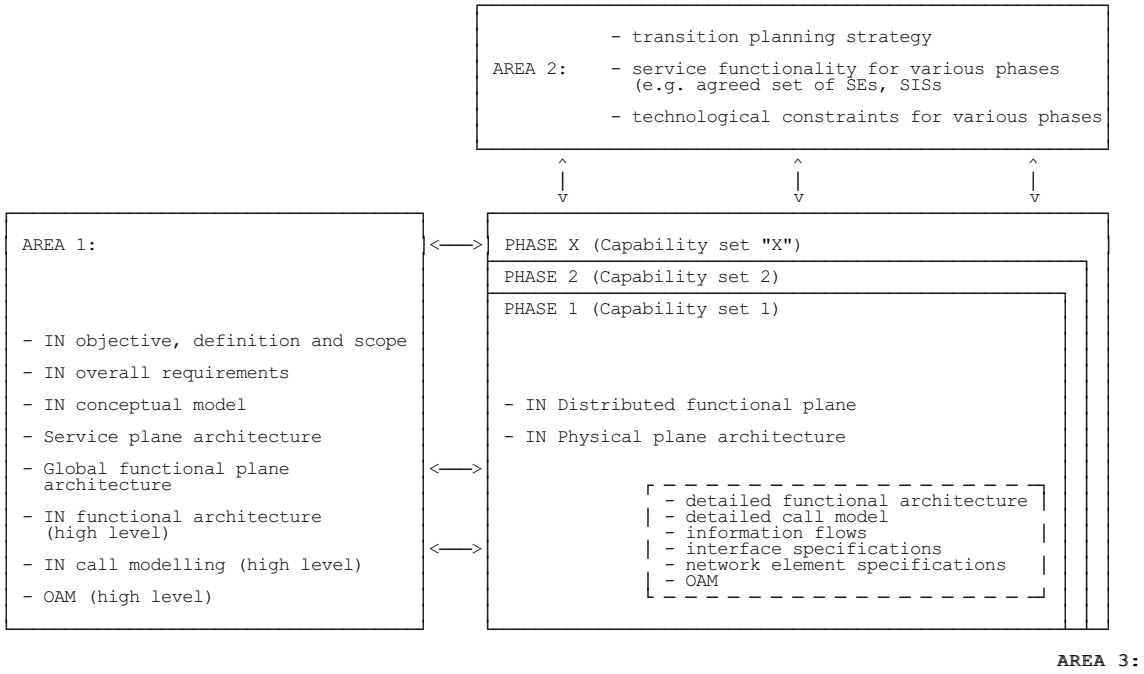
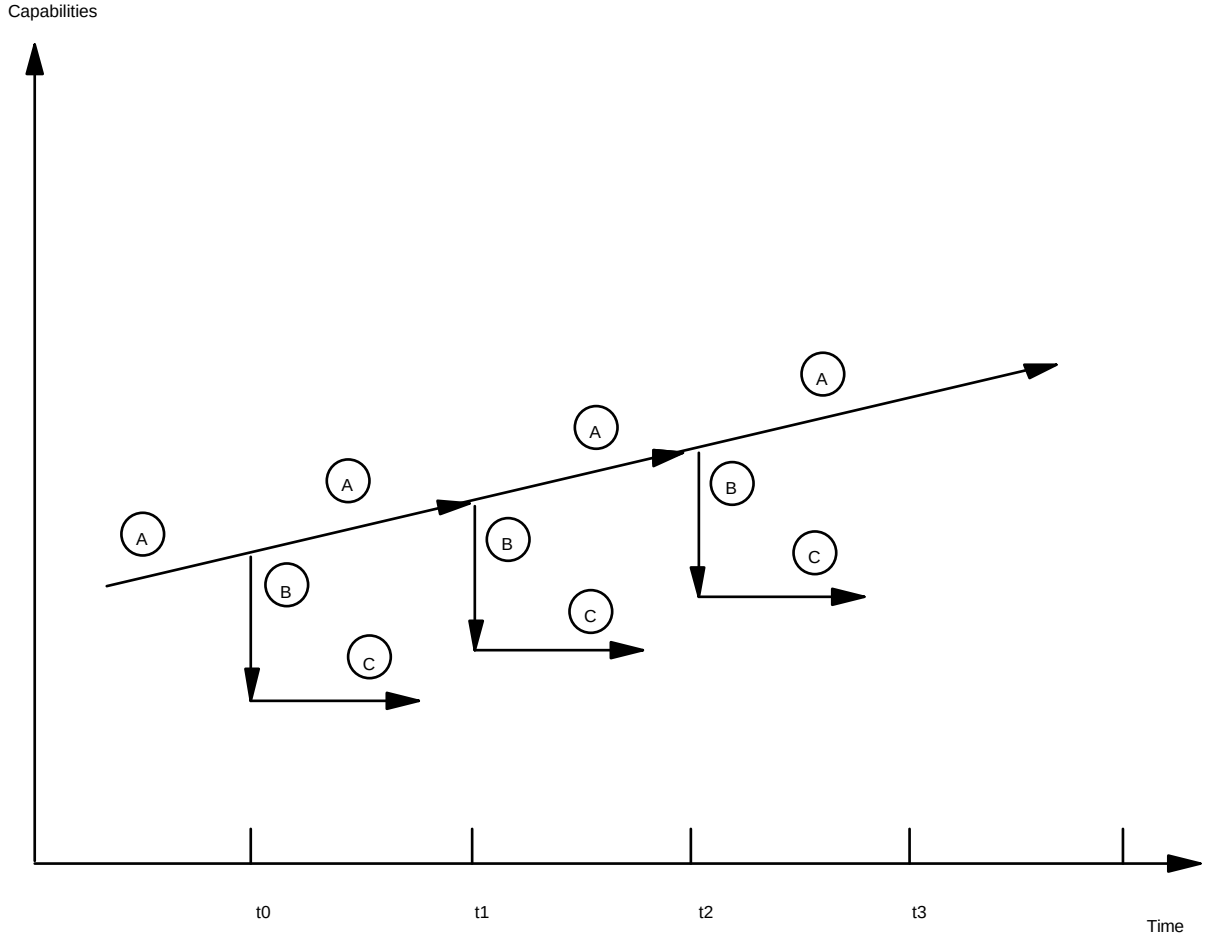


Figure A.2: Refinement of standards area

A.1.5 Phased standardisation

The ultimate IN is an evolving target, therefore in order to take full advantage of the technological possibilities at a given point in time it is necessary to define specific phases from one current view of the target.

The working procedure is shown in figure A.3.



Task A: Concepts and modelling (Area 1)

Task B: "Ease Back": definition of next CS (subset of LTCS) (Area 2)

Task C: Standards for capability Set X (Area 3)

Figure A.3: IN Capability Sets and concurrent work on Long Term Capability Set (LTCS)

Working procedure

The development of IN ETSs can be split into 3 different, consecutive tasks:

Task A: defining an architecture and a comprehensive set of capabilities for the target, long-term IN, as far as current knowledge allows;

Task B: defining a subset of capabilities, to be further detailed, for each phase of IN, consistent with the architecture defined in Task A;

Task C: detailing the capabilities identified in Task B, in order to produce implementable standards.

The consequence of the proposed way of IN-standardisation is:

[1] Although phases cannot be defined beforehand, an implicit phased strategy is still used (phases can be concluded afterwards).

[2] Backwards compatibility can be guaranteed.

The following IN standards/stages are in principle, super sets of their predecessors.

NOTE: The TR "IN Framework" relates to Area 1.

The TR "guidelines for IN Standards" relates to Area 2.

The means for identifying the appropriate documentation for the specific phases of Area 3 is FFS.

Annex B (informative): Overall call modelling

Call Model

Consideration must be given to developing a Call Model which encompasses both the Service Control and the Connection Control Models.

Such a call model is outside the scope of the connection control model. However, this annex contains some call scenarios which the call model must describe, in order to indicate the complexity of the problem.

Scenario 1: This shows a call in which services are invoked which are controlled independently of each other in the Service Control domain. In each case the service is realised by a single instance of service logic opening a single connection control socket to the SSF. Any interactions between the services (desirable or undesirable) are handled entirely within the SSF. Note that this scenario also contains switch based services. In the case of these services the corresponding feature segment will not support a socket to the SCF.

Scenario 2: This shows a call in which an instance of service logic operating through a connection control socket to an SSF, requests the creation of a leg to an end user on another SSF. This illustrates the fact that a leg can be viewed as a network wide object, and the Associator which subsequently creates the leg can also be viewed as providing a network wide function. In this way, the service logic is offered a "global" view of the network.The associator can be regarded as a "global" entity which may encapsulate further associators used to associate together the network trunk circuits involved in the connection through intermediate nodes.

Scenario 3: This shows a call in which a single instance of service logic controls the features of more than one end-user by opening multiple connection control sockets into the SSF. In addition it shows that in support of the execution of this service the SCF has also opened a socket to another functional entity.

Scenario 4: This shows a call in which more than one service is invoked, and where the services co-operate within the Service Control domain. It also shows that the SCF has opened a socket to an SRF.

Scenario 5: This shows a call incorporating aspects of each of the previous four scenarios, including switch based services.It should be noted that the "rules" of the Connection Segment Relationship Model of the CCM will put constraints on the allowable configuration of Connection Segments (and the end-users will only be capable of handling a limited number of services simultaneously).

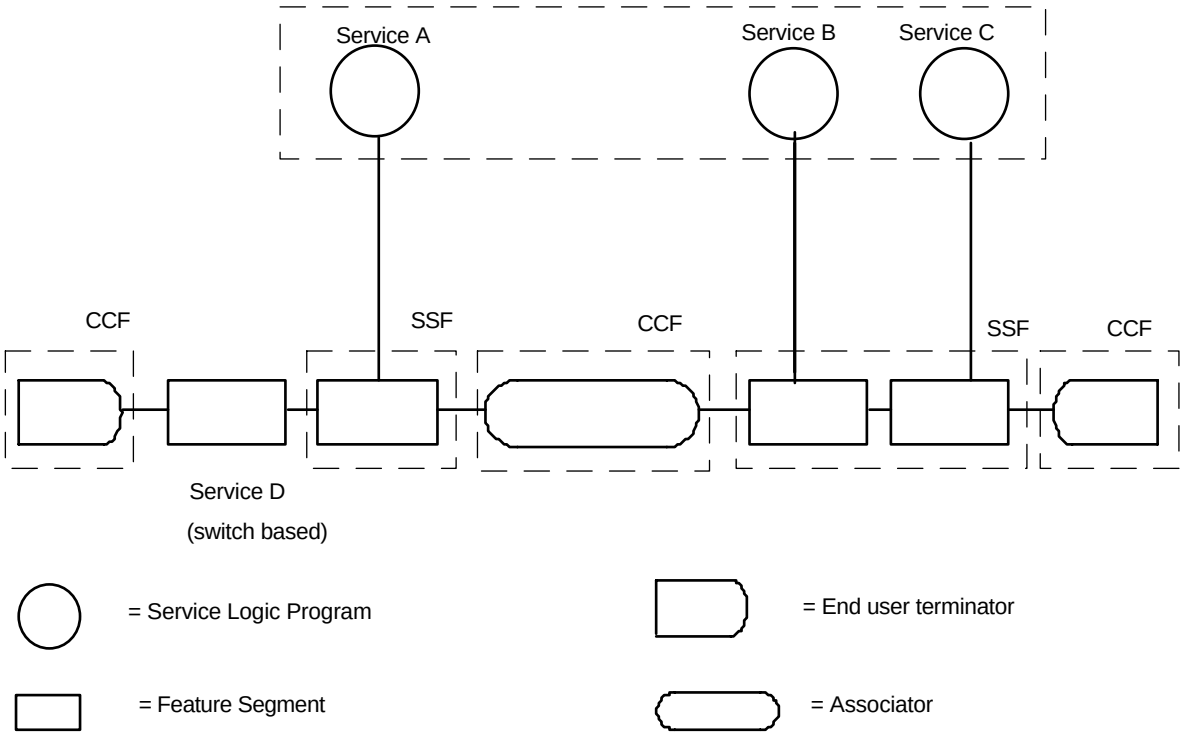


Figure B.1: Scenario 1

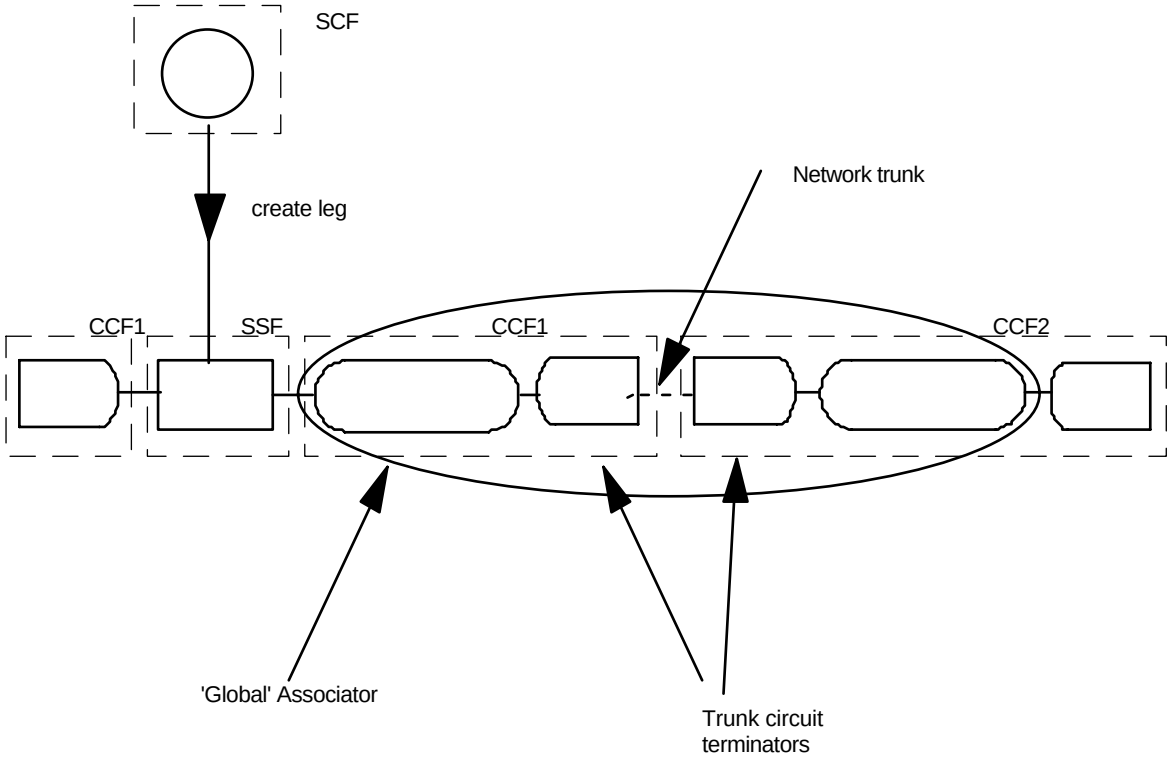


Figure B.2: Scenario 2

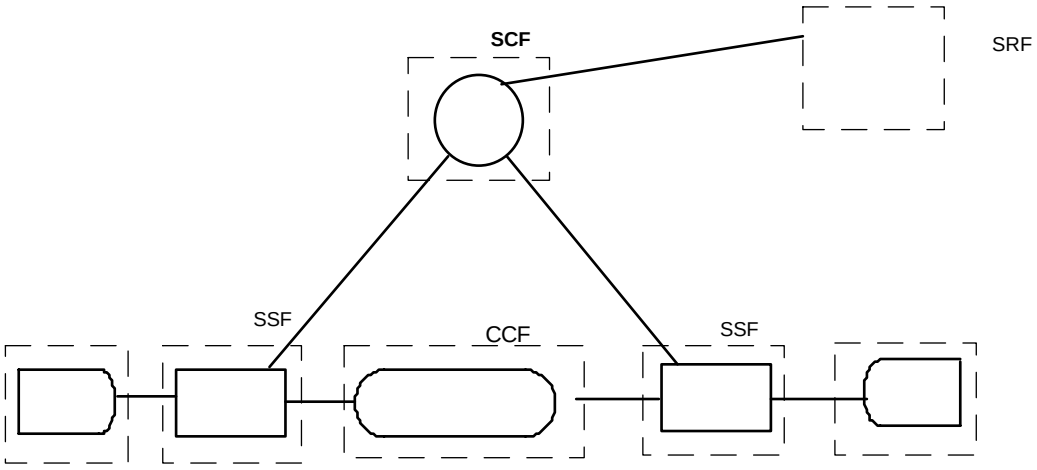


Figure B.3: Scenario 3

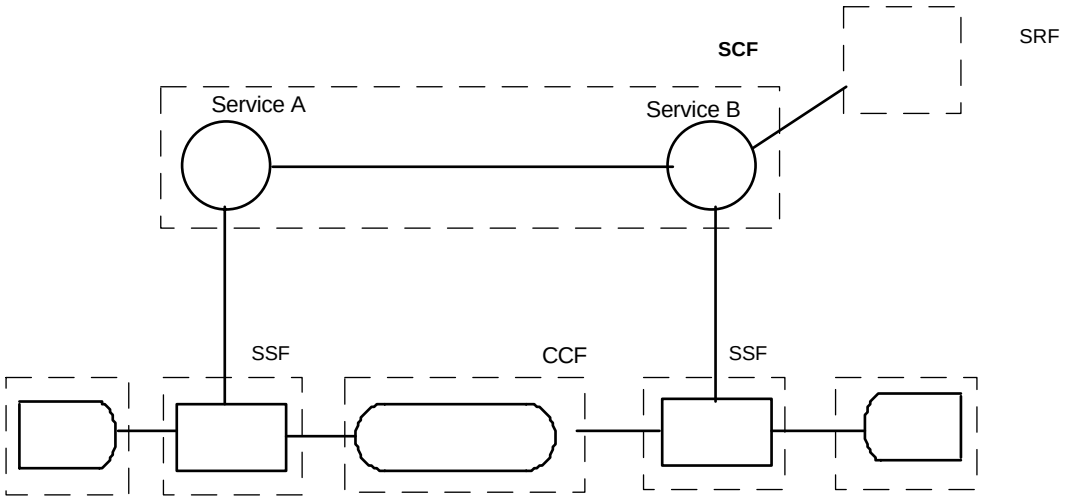


Figure B.4: Scenario 4

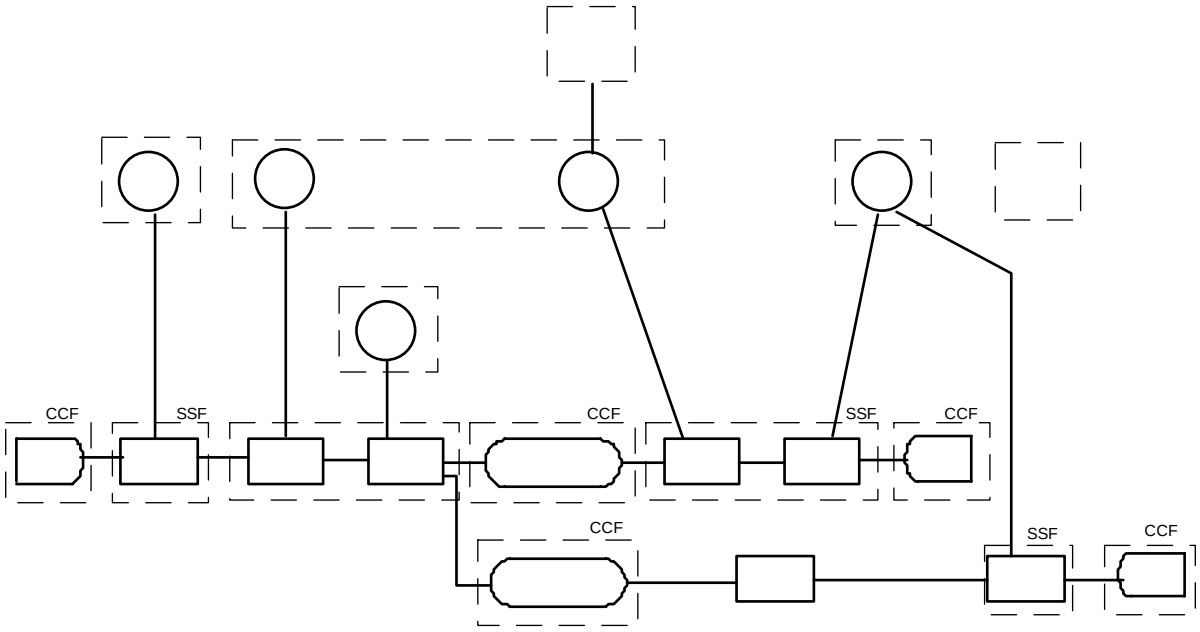


Figure B.5: Scenario 5

Annex C (informative): Connection control model

C.1 General

The objective of this clause is to define functional requirements for connection control as viewed by the service control function. It does not constrain implementation of switches, service control entities or other entities in the network involved in the call.

The connection control model, as viewed from the service control function, is network independent and represents only a logical view of those logical objects that need to be known by the service control function. It does not represent the physical configuration in the SSF.

It is worth noting that, even with another model, the same SSF-SCF interaction could have been described.

C.2 Connection control modelling

C.2.1 Environment

In order to describe the nature of the connection control model, first it is necessary to define the various entities associated with the environment in which it resides.

An end-user accesses the network via a terminal and views that network as a "black box" system. A call may be defined as an arrangement providing a relationship either between two or more end-users, or between an end-user and the system (i.e. network) for the purpose of communicating information.

NOTE: An end user may be either a person or a machine.

A call may span several switching nodes in the network. The end-users can be parented on different switching nodes. Thus the call reflects the network wide view of the relationship. Individual switching nodes may only be aware of part of the call, and some of the end-users involved.

Calls can be initiated by end-users directly or by the network on behalf of end-users. Additional capabilities (services), can be invoked by end-users to supplement basic calls in order to maximise the effectiveness of the call relationships. A single call must support the invocation of multiple capabilities either by the end-users involved, or by internal action by the network.

Since individual switching nodes may only be aware of part of the call, they may also only be aware of some of the capabilities invoked on a call. It is desirable therefore, that the capabilities invoked by a calling user on one switching node, can operate independently of capabilities invoked by a called user on another switching node. This should also apply when the calling and called user are on the same switching node.

In addition it is desirable that multiple capabilities invoked by a single end-user on the same call can operate independently of each other, and that capability independent mechanisms can handle interactions (both desirable and undesirable ones) between different capabilities, regardless of whether these capabilities are IN based or switch based. Furthermore it must be assumed that non-IN capabilities (e.g. existing switch based features) will continue to be developed in the network.
IN based capabilities will be realised by the execution of service logic residing in the Service Control Function, (SCF) which will invoke capabilities provided by other functional entities. Principal among these will be the Service Switching Function (SSF).

The SSF corresponds to the call processing functionality resident in the switching nodes of the network, and is primarily responsible for the provision of connections (i.e. communication paths) between the end-users.

It is necessary to find some way of describing the relationship between the control function (SCF) and the switching function (SSF). This is the purpose of the IN Connection Control Model (CCM), which is described in this subclause. Several points should be noted concerning this model.

- It does not assume any physical architecture. Thus, the IN service logic, including that acting on a single call, may reside in several geographically diverse locations.

- It defines the relationship between all SSFs and all SCFs involved on a particular call. A particular SCF only has visibility of the part of the call visible through its socket in to a particular SSF. Thus there is no CCM at the SCF domain level.

- It may span several SSFs, but each SSF only has visibility of that part of the CCM relating to it and the services acting on it (these may be from several sources).

- It provides mechanisms to support the assumptions listed above concerning the independence of operation of different capabilities.

- It is only intended as a way of describing the relationships between the SCF and SSF, and is not meant to imply how such a relationship may be implemented in physical terms.

C.2.2 The socket

It is assumed that a service, that is a particular instance of service logic residing within the SCF, will consist of a number of separate invocations of capabilities or features, provided by several different network functions residing in different Functional Entities. e.g. SSFs, database, Intelligent Peripherals (IPs), etc. In some cases, a feature may also be a complete service in its own right. E.g. the capability of providing a three party call may be either invoked as part of another service or may be used as a service itself.

In order to invoke these features, the service logic must open "sockets" into the appropriate network functions, i.e. a dialogue must be started between the two Functional Entities concerned.

The socket is a "window" which may be "opened" at the start of the dialogue between the two functions and "closed" at the end of the dialogue. For modelling purposes it need not be modelled as an object in the software engineering sense.

Each socket (or window) offers the service logic limited visibility of the network function consistent with the nature of the feature being invoked. That is, service logic acting through a socket may only issue requests which are consistent with the reason for which the dialogue was established.

Sockets may be opened and closed at various times during the lifetime of a call.
There are various types of socket, corresponding to the different reasons for which a dialogue has to be established between two functional entities. These will be discussed in more detail later. Every socket is an instance of a particular socket type. Multiple sockets of the same or different types may exist concurrently related to a single call, that is, multiple dialogues may be being carried on at the same time. These sockets (dialogues) could all have been opened (started) by the same instance of service logic, or by different instances.

The CCM is only concerned with those types of sockets which may be opened between a SCF and a SSF, i.e. it is only concerned with describing the dialogues which may take place between a SCF and a SSF. These sockets may be opened either by the SSF as a result of triggering or by the SCF. (That is, a request for a dialogue may be initiated by either the SCF or the SSF).

As an example, in a simple number translation service, the SSF detects the dialled digits (e.g. 800). It will now open a socket (start a dialogue) with the SCF to obtain the necessary routing information.

Within the context of this socket, a "provide instructions" message is sent to the SCF, and the SCF responds with the new routing information. In this case, the response message ends the feature, the transaction and the socket. A feature requiring extended interaction (e.g. Call Waiting) can be viewed as being implemented by a somewhat longer-lived socket.

Some complex features may require access to more than one existing call via the same socket (e.g. Call waiting). In some such cases there may be more than one connection point within the Socket (see subclause C.2.6) and potentially these may be "merged" to form a resultant single call.

C.2.3 Multiple sockets

An important part of the connection control model is the ability to manage multiple features (sockets) being active on the same call as viewed by the SSF. Thus, the model must incorporate mechanisms for handling possible interactions between features. These mechanisms must be able to handle interactions between switch based features, between switch based and IN features and between IN features. However, the possibility of interactions and relationships between IN features being managed internally within the SCF domain (i.e. external to the CCM) is not ruled out. (See subclause C.2.11 for further details on feature interaction).

C.2.4 Logical objects and socket types

The view in to SSF functionality which is offered to the SCF via a socket is defined and scoped in terms of the logical objects which are accessible, and the operations which can be performed on those objects. These objects represent an abstraction of resources contained within the SSF which the service logic can manipulate. According to the purpose for which the socket was opened (e.g. connection control, data management) there will be different logical objects visible through the socket.

There are different types of socket which are differentiated from each other according to the types of objects accessible and the range of operations allowed. In this way a socket provides a bounded scope of visibility and influence into call processing activities.

Possible types of socket which may be opened between a SCF and a SSF are listed below. These correspond to the different reasons for which a dialogue may be established between the SCF and the SSF. Examples of objects which are accessible through these sockets are given in brackets, where known.

- connection control (legs, connection points);
- data management and information collection (counters, buffers, trigger table entries, etc.);
- status monitoring (end-user lines, queues);
- traffic management (traffic management filter);
- resource control (DTMF receivers, tone recognition devices, etc.).

It may be necessary to introduce the concept of socket sub-types. Sub-types of sockets will be more constrained with regard to the numbers of objects available in the socket and the operations which can be performed on them. For example, a service which only monitored a call and calculated a charge would operate through a sub-type of the connection control socket which did not permit it to create legs, and only provided one connection point object and two leg objects. A three party service would operate through a sub-type of the connection control socket which only allowed three legs to be present at any one time in the socket.

C.2.5 Components of the connection control model

In order that the connection control model can correctly define the relationship between the SCF and SSF, there are three things which it must describe. These can be regarded as separate but inter-related components of the model:

- It must describe how the basic call processing of the SSF detects trigger conditions and opens a socket to the SCF. (See Clause C.10 for a fuller description of trigger points and trigger conditions).

- It must describe the interaction between the SCF and the SSF which takes place within the bounds of the connection control socket type in support of feature operation.

- It must describe the relationships between multiple connection control sockets controlling separate features acting on the same call, and between these sockets and basic call processing functionality contained in the SSF.

It is proposed to call these three components the Basic Call Model, the Connection Control Socket Model, and the Connection Segment Relationship Model.

NOTE: Other models may be required to describe the relationship between multiple sockets of other types.

The requirements on the Basic Call Model are described in Clause C.10.

It is the third component, the Connection Segment Relationship Model (CSRM), which is key to the overall shape of the complete connection control model.

On a single call, there may be several sockets concurrently open between the SCF and SSF to do with the execution of a particular feature. Each socket exposes a set of logical objects which only have meaning within the context of that dialogue. It is the responsibility of the CSRM to provide a means by which operations requested on different objects in different sockets can be co-ordinated. (i.e. it must provide a mechanism for managing feature interactions). Interactions, both desirable and undesirable, must be managed as far as possible in a feature independent manner, and potential conflict situations at the mapping of logical to physical resources must be avoided.

Before describing the CSRM, it is first necessary to describe in more detail the connection control socket type. This is the most important socket type employed across the SCF-SSF interface, and is also the one which potentially is most susceptible to interaction problems.

The CSRM only describes the relationships between sockets of the Connection Control type.

C.2.6 The connection control socket model

The connection control socket allows access to two types of logical object, a leg and a connection point. These two objects can be used to provide a logical representation of the state of a connection (call) at any one time. More than one of each type of object can be accessed through a single socket. Objects of both types may be created and destroyed within the lifetime of a socket. The state of a connection control socket at any point in time can be described in terms of the objects of which both the SCF and SSF are aware. This can be represented pictorially. Figure C.1 shows a "snapshot" of a connection control socket at the point where the SSF and SCF are both aware of 3 legs and one connection point object.

A leg is an object visible through a connection control socket which represents the existence of a logical communications path towards some addressable logical entity in the network, e.g. an end-user, or an information source or sink. It has no meaning outside the context of a connection control socket (dialogue). If there is a leg object present in the socket, then from the SCF's point of view there is a direct communication path to the user on the end of the leg. The presence of the leg object gives no information as to whether the communications path it represents actually spans one or several network nodes.

Specific attributes may need to be attached to legs. Some attributes may be determined by the SSF (e.g. leg on which triggering occurred) whilst others may be determined by the SCF (e.g. leg associated with the end-user currently controlling the feature). This area requires further study.

There may be restrictions on the number of legs accessible through a particular connection control socket (achieved via the socket sub-type concept). All legs must be uniquely identified within the context of a socket. Leg identifiers may be assigned either by the SSF or SCF.

A leg can be in one of a number of states. These are for further study, but a minimum set of two states is required:

- joined (connected to a specified connection point);
- disjoint (not connected to a connection point).

Additional context information may be associated with a leg, e.g. the logical entity to which the leg extends, or the event management mode. Operations will exist enabling information to be sent to and received from the end-users accessible via the legs.

A connection point is an object visible through a connections control socket representing a logical bridge. If two legs are joined to the same connection point, then this indicates that within the context of this connection control socket, the users on the ends of the two legs can communicate with one another. Note that a second feature acting via another socket may have split the logical communications path, causing the end-to-end logical communications path to be broken, but this is not known by the first feature. There may be constraints on the number of legs which can be joined at a given connection point within a particular sub-type of the connection control socket. In addition there may be constraints on the number of connection points allowed. All connection points must be uniquely identified within the context of a specific socket. Connection point identifiers may be assigned by either the SSF or the SCF.A connection point has states corresponding to the number of legs currently connected to it Additional context information may be associated with a connection point such as the maximum allowed number of connected legs.

When a connection control socket is created as a result of a trigger condition detected in the SSF, a leg/connection point configuration will already be present in the socket. This will reflect the state of the connection existing at the time the socket was opened. If a connection control socket is created at the request of the SCF, then there will be no leg/connection point configuration present in the socket. The operations which may be performed on legs and connection points are listed in Clause C.5 of this annex.

C.2.7 Virtual switch concept/connection segment

Embodied in the connection control socket model described above is the concept that every independent feature within the complete Socket Relationship Model (CSRM) for a call has direct access to its own private virtual switch solely under its control. This virtual switch is represented by the leg and connection point objects. Each of these virtual switches is modelled by a connection segment within the CSRM. The socket can be viewed as a control interface into the connection segment from the SCF. A complete call involving multiple, simultaneously active features comprises a number of connection segments linked together serially to form a single chain or a configuration of inter-connecting chains.

Figure C.6 shows an example chain of 6 connection segments of various types. Figure C.7 is an example showing interconnecting chains.

In many ways the chaining of separate virtual switches (connection segments) of the CSRM within a switching node emulates the interconnection of physical switches when a basic call is routed from node to node across the network.

There are various types of connection segments. The connection segments specifically supporting feature control are called feature segments. It is worth noting that switch based features can also be modelled by feature segments. However they would not possess a socket interface to the SCF. Their logical leg and connection point objects would instead be controlled internally by the call processing logic based in the SSF.

An individual feature segment is unaware of other feature segments which may exist within the call. As well as the control of features, basic call functionality is also represented in the CSRM by connection segments. There are two types of basic call segments: terminator segments and associator segments.

Connection segments of any type are joined together by means of links. The presence of a link between two segments implies that there is a signalling path established between them over which messages can pass.

The signalling capability on a link can be regarded as a superset of the circuit-related signalling for external circuit interfaces (e.g. CCITT Recommendation Q.931, L/D, CCITT 7 ISUP). Additional signalling information beyond that carried for external circuit-related signalling may also be carried along these links.

C.2.8 Basic call segments

There are two types of basic call segment. A Terminator Segment represents the external access point of a logical transport path into the switching node (e.g. a subscriber line, or trunk circuit), and the basic call processing functionality associated with dealing with originating or terminating calls from/to this access point. In this sense, terminators can act as either the incoming (e.g. calling party) or the outgoing (e.g. the called party) end of a call. In some cases a single terminator acts in both incoming and outgoing roles (e.g. party B in a multi-way calling case where A calls B, and then B adds on C).

A terminator segment can be regarded as controlling a virtual switch consisting of two legs and a connection point. One leg of the switch represents the external point into the switching node. The other leg represents the logical communications path to the other end user. The logic controlling this virtual switch resides wholly within the terminator segment (i.e. it resides only in the SSF).

A terminator segment may be linked to either an associator segment (as in a basic call) or to a feature segment. A terminator segment is shown in figure C.2.

An Associator segment represents the association between two end-users of the network, and the call processing functionality concerned with connecting end users together (e.g. such things as the decoding of dialled digits and routing). As associator is required within any chain of segments connecting the terminators corresponding to two different users. The simplest definition of "different users" is that from the perspective of the CSRM different users correspond to different access points on the switching node. Even if a particular terminator corresponds to a trunk circuit it can still be viewed as "leading to" an end-user.

An associator controls a virtual switch composed of two legs, corresponding to the calling and called parties, and a single connection point. The "logic" which controls this virtual switch resides wholly within the associator segment and reflects the rules of the network, whereas the logic which controls the virtual switch within a feature segment reflects the rules of the service. [For example, the point in the call at which the two legs are joined at the associator's connection point is dependent on such rules. It might be at the point of outgoing channel seize, on receipt of address complete, or upon answer, and could be at different times for the forward and backward transmission paths].

An associator can be linked to terminator and/or feature segments. An associator is never directly connected via link to another associator. In other words, calls with more than two parties involved always include a feature segment to manage the forwarded, transferred or multi-way connection.

An associator segment is shown in figure C.3.

C.2.9 Feature segment

A feature segment is a specific type of connection segment which supports a connection control socket interface for the control of a feature by the SCF.

It provides the necessary mechanisms to translate a request of an operation on an object within a socket into the invocation of the appropriate functionality provided by the SSF, and enables the segment to interface correctly with other connection segments involved in the call, thus making up the complete call topology.

A socket cannot exist unless there is a feature segment in place in the CSRM to support it.

A typical feature segment is shown in figure C.4. The diagram shows that at this point in time, the SCF and the SSF are both aware of three legs and a connection point. An alternative representation is shown in figure C.5.

The testing for the trigger conditions which have to be satisfied before a particular feature is invoked resides within the terminator and associator segments.

When a trigger occurs and a new feature segment is created to support the connection control socket (dialogue), there are rules relating to where this new segment should be positioned in the existing segment chain relative to the basic call segments and to other feature segments already active (see subclause C.2.11).

Feature segments can be positioned in the parts of chain extending between a terminator (either incoming or outgoing) and an associator.

Thus every feature segment has a terminator end and associator end. The terminator represents the end-user to which the feature is affiliated. The higher the precedence of the particular feature with respect to an end-user, the closer will it be positioned to the corresponding terminator in the chain.

Signals originating at the terminator end of the feature segment will be reported to the SCF via the associated connection control socket as originating from the controlling leg.

The associator end of a feature segment can be linked to other associator segments (either directly or via other feature segments). Associator segments can be created either as part of a basic call, or as a result of feature operation. In the latter case, a create leg operation by the feature logic will cause a new link to be created to a new associator segment.

Figure C.6 shows a chain of segments corresponding to an example 2 party call with features active. Examples of chains for more complex calls involving features are shown in Annex D.

The links between the segments in a chain can be likened to a "signalling pipe" inter-connecting the incoming and outgoing sides of the switched node, down which forward and backward signals can flow. In order to place a segment in a chain, an existing link between two segments must be broken, and the feature segment inserted into the gap. When this is done, signals which would have passed directly between the two segments must now pass through the feature segment. In this way, the feature segment can tap into the pipe taking action only on those signals which are relevant to it, and being transparent to the remainder. Segments closest to the terminator will have precedence with respect to the processing of signals originating from the end-user represented by that terminator. Thus the four signals management modes (transparent, duplicate, intercept and ignore) can be supported by each feature segment. Event management and monitoring is described further in Clause C.5 of this annex.

When the service logic is completed and the dialogue pertaining to the socket model is terminated (normally by the SCF), the feature segment removes itself from the chain leaving an unbroken link.

However the feature segment can take a more active role in support of the feature operation. It can police the socket interface, making sure that the service logic in the SCF keeps within the bounds of the socket model. In addition when the service logic completes, the feature segment can check that no undesirable "debris" has been left behind, such as disjoint legs. Any such debris can be removed before the feature segment removes itself from the chain.

C.2.10 Separation of logical and physical connectivity

In order to support multiple features acting on a single call, it is a requirement to be able to determine unambiguously the physical connectivity resulting from the separate logical connectivity viewpoints relating to each connection segment.

Each connection segment corresponds to a private virtual switch. The actions of joining and splitting legs via connection points within each connection segment is a purely logical representation of the connectivity.

The SSF must ensure that the correct actions take place in response to operations requested by the SCF on leg objects within the context of a specific connection control socket. As part of this activity the SSF must handle contention between different sockets for the same physical resource.

The algorithm to do the mapping from the logical connectivity configuration represented in the CSRM to the physical connectivity configuration is not in itself part of the connection control model, but instead can be viewed as the function of a "connectivity server" forming part of the embedded function of the switching node.

Figure C.6 shows the mapping performed by the connectivity server for the example two party call.

Figure C.7 shows the mapping for a more complex call involving four parties and multiple features.

C.2.11 Feature interaction management

The CCM must provide mechanisms for managing interactions between multiple features acting on a call. Note that this consists both of allowing desirable interactions and preventing undesirable interactions. As stated in subclause C.2.3 of this annex, interaction between switch based features, IN and switch based features and IN features must all be handled.

As discussed earlier, interaction could be managed in the SCF domain, the SSF domain or both. There is a body of opinion which believes that interaction problems are potentially more difficult to solve the further one moves away from the switching system which is the source of the external end-user events. At this stage in the development of IN standards, it is probably best assumed that interactions should be handled within the SSF where it is practical to do so, but that there will be some types of interaction particularly those associated with advanced, complex services which can only be handled within the SCF domain. So work should proceed on the basis that both techniques are appropriate.

Regardless of where feature interaction management is carried out, it can be done by either of two basic methods - a specific feature by feature interaction method, or a general rule-based method.

The specific method relies on specifying each possible interaction for every possible combination of features. As the numbers of features gets large, (which is one of the aims of IN), this specification quickly becomes complex, and complicates the tasks of the feature writer. Furthermore, as each new feature is added, its many possible interactions must be decided and specific data must be entered into either the SSF or the SCF to specify how each interaction is to be resolved.

The general method relies on a set of feature interaction rules and specific data characteristic of each feature. Applying the rules to the feature characteristics results in precedence decisions amongst the features. This method is clearly more desirable, since the data needed to specify feature interactions is much less, and adding a new feature to the system requires only assigning the proper characteristics to the feature. More importantly, it enables the feature writer to design a new feature independently of other features which may already exist.

The connection control model provides two important characteristics in support of the general method of feature interaction management.

Firstly, the socket concept ensures that features acting on a call operate without any knowledge of any other features acting on the same call at the same time.

Secondly, the CSRM provides a way of "chaining" the segments, relating to different features, together, in order to provide the "best" overall service to the end-users. The order in which the segments are chained together dictates the feature precedence in terms of their response to external events, and of their impact on the overall logical connectivity.

The placement of feature segments in the chain is carried out according to the following procedure.

Assume there is an already existing chain of feature segments. A new feature is now triggered and it must be decided whether it is compatible with the existing set and if so where in the chain the new segment can be placed.

If the event which caused the trigger is associated with an end user then this process should start from the terminator end of the chain. If the event is from the network (e.g. answer from a far party), then the process should start from the associator end. The new feature is now "compared" with the feature existing at that point in the chain. This comparison could have one of three possible results.

- If the new feature is incompatible with the existing feature then either the new feature must be rejected, or it must replace the existing feature;

- If the new feature can take precedence over the existing feature, then it will be inserted into the chain "before" the existing feature. Note that if a new feature's incompatibility with lower precedence features must be accommodated then it will be necessary to search the whole chain for incompatible features before inserting the new feature;

- If the new feature is not incompatible, but cannot take precedence over the existing feature then the new feature should be compared with the next feature in the chain and so on until no features are left (i.e. the associator or terminator is reached). Provided the feature has not been found to be incompatible with any previous features then by default, it will be placed next to the associator or terminator.

This comparison of precedence between two features would be performed by a Feature Interaction Management Function, residing within the SSF, and would take place according to certain rules, based on the known characteristics of features involved. An example of this might be that non-persistent features (those that run to completion before another feature is triggered) should take precedence over persistent features. Other rules can be derived. While these rules may not be sufficient in all cases to decide precedence they do at least reduce the problem considerably compared with the specific feature by feature method.

In addition to deciding where to place a newly feature segment in the chain, the precedence between features which may be triggered at the same trigger check point must also be decided. Again, rules can be applied to produce a precedence ordered list of these features. An example of a rule is that those features which are trying to hinder the progress of a call must take precedence over those features which are trying to progress a call.

The data which characterises a particular feature can be provided at feature provisioning time (e.g. in static data, such as the trigger table entry), or at feature run time.

The foregoing has assumed that feature interaction management is carried out within the SSF. If it is decided that it is better carried out in the SCF then the same rule based methods should apply. In this case it would probably be necessary for the SCF to maintain a representation of that part of the CCM relating to the SSFs under its control, in order that it can construct its own CSRM.

For very complex services it may be necessary to have a single point in the network (e.g. a particular SCF) which maintains a complete representation of the CCM for a particular call, however this is outside the scope of this document.

C.2.12 The relationship between the basic call model and the connection segment relationship model

A key characteristic of the CSRM is that all active entities which can impact on the connectivity configuration of the call must be reflected within the connection segments which make up the particular CSRM configuration. This applies to both feature and basic call functionality.

The Basic Call Model describes the sequence of activities needed for the execution of a basic call between two end-users. It is divided into three parts corresponding to the functionality represented by the basic call segments of the CSRM (i.e. Terminator - Associator - Terminator). Its main purpose is to identify standard trigger check points in the basic call sequence.

Upon triggering at such a point, trigger conditions have to be checked. If the result of this check is positive, then the sequence associated with the basic call segment (e.g. an associator) in which the trigger occurred, is temporarily suspended, the appropriate feature segment is created and is positioned correctly into the segment chain. Depending on what operations the SCF subsequently invokes via the socket model the SSF may choose to either discard the suspended basic call segment (it being consistent with the feature being controlled), or to resume within it at an appropriate point, normally close to or at the point of suspension. Thus the specification of a resumption point into the basic call model does not require to be explicitly identified by the SCF via the socket model.

C.3 Definitions relating to the connection control model

The concepts of prime importance which have been introduced in the Connection Control Model are defined below.

Connection control model

The Connection Control Model is a model that describes the static and dynamic aspects of the interaction between the SSF and SCF in order to co-ordinate the call processing activities of the SSF in support of IN services.

The Connection control model is a unified model comprising three components:

a) The basic call model

b) Connection control socket model

c) Connection segment relationship model.

Basic call

A basic call is a call between two end-users that does not require features (e.g. a POTS call).

Basic Call Model (BCM)

The Basic Call Model describes the sequence of activities needed for the execution of a basic call. The BCM is part of the Connection Control Model. It is divided into three parts corresponding to the functionality represented by the basic call segments of the CSRM (i.e. Terminator - Associator - Terminator). Its main purpose is to identify standard trigger check points in the basic call sequence.

Connection Control Socket Model

The Connection Control Socket Model is part of the Connection Control Model and models the interaction which takes place between the SCF and the SSF within the bounds of a Connection Control Socket type in support of a feature operation.

Connection Segment Relationship Model

The Connection Segment Relationship Model is part of the Connection Control Model and describes the relationships between the basic call processing functionality and multiple features acting simultaneously on a call. Each IN based feature is controlled by the SCF via a separate instance of a connection control socket supported by a feature segment.

Feature

A feature is a re-usable unit of one or more service capabilities forming all or part of a service (IN or switch based). With respect to the IN connection control model a feature is controlled by the SCF via a single socket instance.

Socket

A socket is an instance of a logical point of access into the Service Switching Function through which service logic pertaining to a single feature invocation can operate. Each socket instance offers the service logic a limited aperture of vision and influence into the call processing consistent with the nature of the feature. A single call may have several sockets existing simultaneously, each one corresponding to a different feature. Sockets may be created and destroyed at various times during a call. Socket may be created either by the SSF or the SCF.

Socket type

A socket type is a particular class of socket with a specific set of properties. Socket types are differentiated from each other according to the types of objects accessible and the operations which may be performed on those objects. Every socket is an instance of a particular socket type. The socket type of prime importance is the Connection Control socket type in which the objects accessible are legs and connection points.

Socket sub-type

The concept of socket sub-types allows constraints to be imposed on a socket type with regard to the numbers of logical objects (e.g. legs or connection points) which are accessible via a socket instance, and to the operations which can be performed on them.

Logical object

In relation to the Connection Control Model, a logical object represents an abstraction of a resource or resources contained within the SSF which service logic can manipulate via a socket instance.

Leg

A leg is an object accessible via a socket representing a logical communications path towards an addressable logical entity in the network, (e.g. an end-user, or an information sink or source). A leg object may be created at the time a socket is opened in order to represent an existing communications path, or it may be created by the subsequent operation of the service logic. All legs within a socket have a unique local identity.

Connection point

A connection point is an object accessible via a socket representing a logical bridge indicating which legs can logically communicate with each other, from the perspective only of the single feature acting through the socket. (Note that a second feature acting via another socket on the same call may have split the logical communications path, causing the end-to-end path to be broken, but this is not known by the first service.) More than one connection point can be accessed via a single socket. A connection point object may be created at the time of socket creation in order to represent an existing connection between two legs, or it may be created by the subsequent operation of the service logic. All connection points within a socket have a unique local identity.

Connection Segment

The Connection Segment Relationship Model is composed of Connection Segments, which can be chained together serially in various configurations. Each connection segment controls its own private virtual switch represented by leg and connection point objects. There are two types of Connection Segment: Basic Call segments and Feature segments.

Basic Call Segment

A Basic Call segment is one type of Connection Segment in the Connection Segment Relationship Model which reflects the functionality of part of a basic call. There are two types, Terminator and Associator segments.

Terminator Segment

A Terminator Segment is one type of Basic Call Segment within the Connection Segment Relationship Model. It represents the external access point of a logical transport path into the switching node (e.g. a subscriber link or trunk circuit) and the basic call processing functionality associated with dealing with originating and terminating call from/to this access point.

Associator Segment

An Associator Segment is one type of Basic Call Segment within the Connection Segment Relationship Model. It represents the association between two end-users of the network and the basic call processing functionality concerned with forming a connection between two end-users.

Feature Segment

A Feature Segment is one type of Segment within the Connection Segment Relationship Model. It may correspond to either an IN based feature or a switch based feature. In the former case, the SCF controls the feature via in instance of a connection control socket type.

Link

Connection segments are joined together serially forming a chain in the Connection Segment Relationship Model by means of links. Each link corresponds to a signalling and logical transport path between two adjacent connection segments.

Chain

A chain is a component of the Connection Segment Relationship Model. It consists of a number of connection segments joined serially by means of links. The CSRM may comprise a single chain or a configuration of inter-connecting chains.

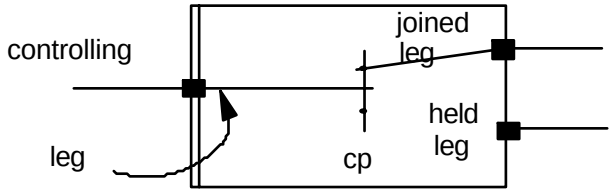


Figure C.1: An example connection control socket (CP = Connection Point)

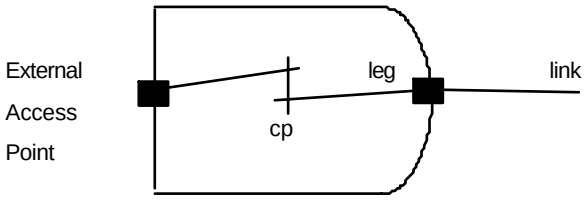


Figure C.2: A terminator segment

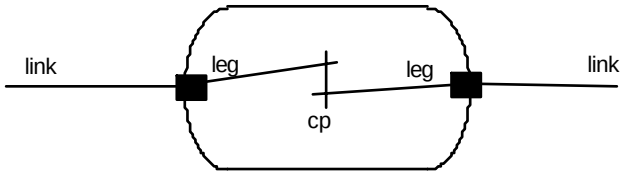


Figure C.3: An associator segment

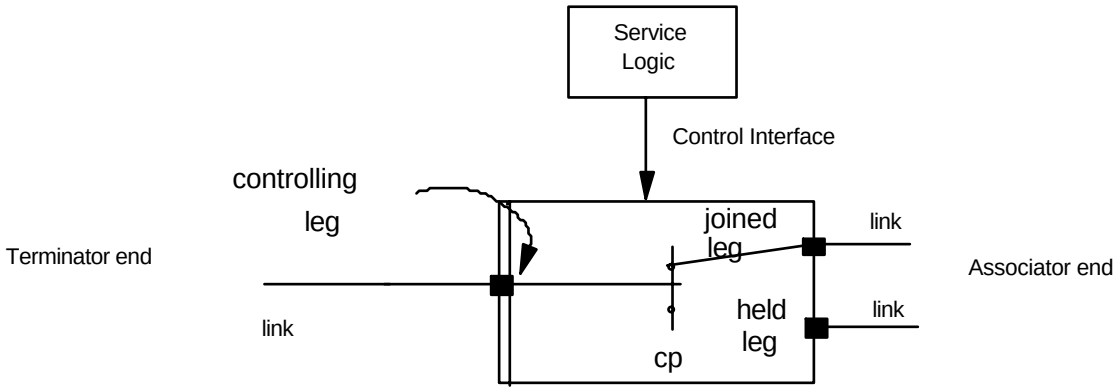


Figure C.4: A typical feature segment

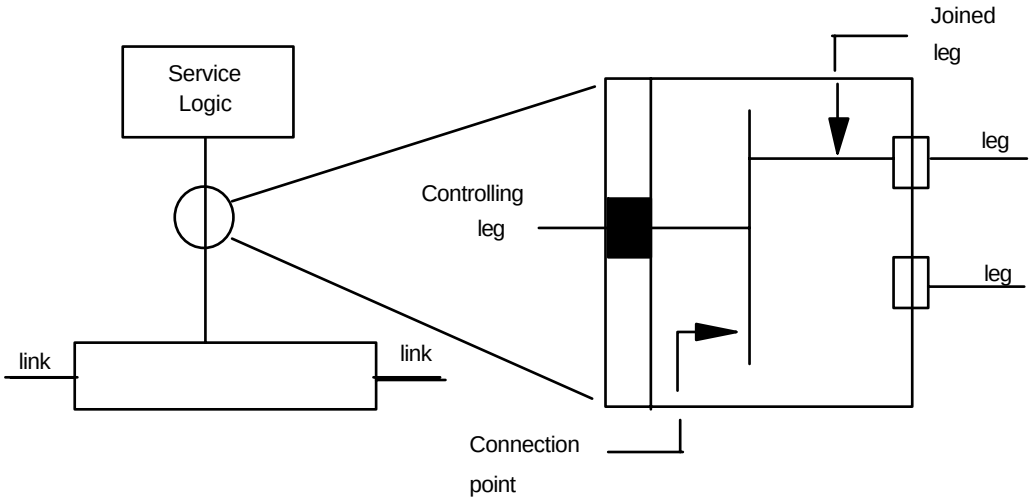


Figure C.5: Feature segment and associated socket

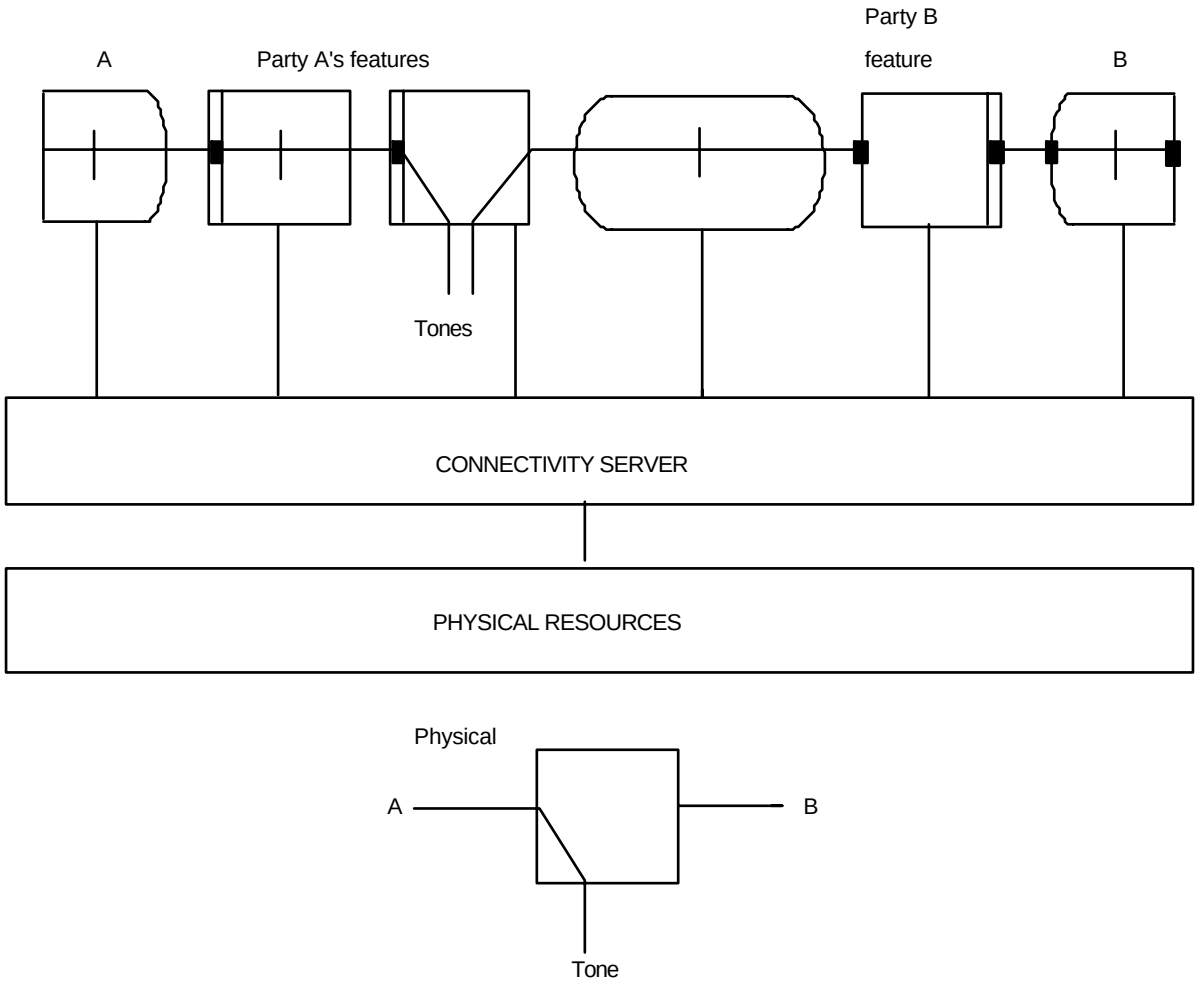


Figure C.6: An Example 2 party call with features active, also showing logical to physical connectivity mapping

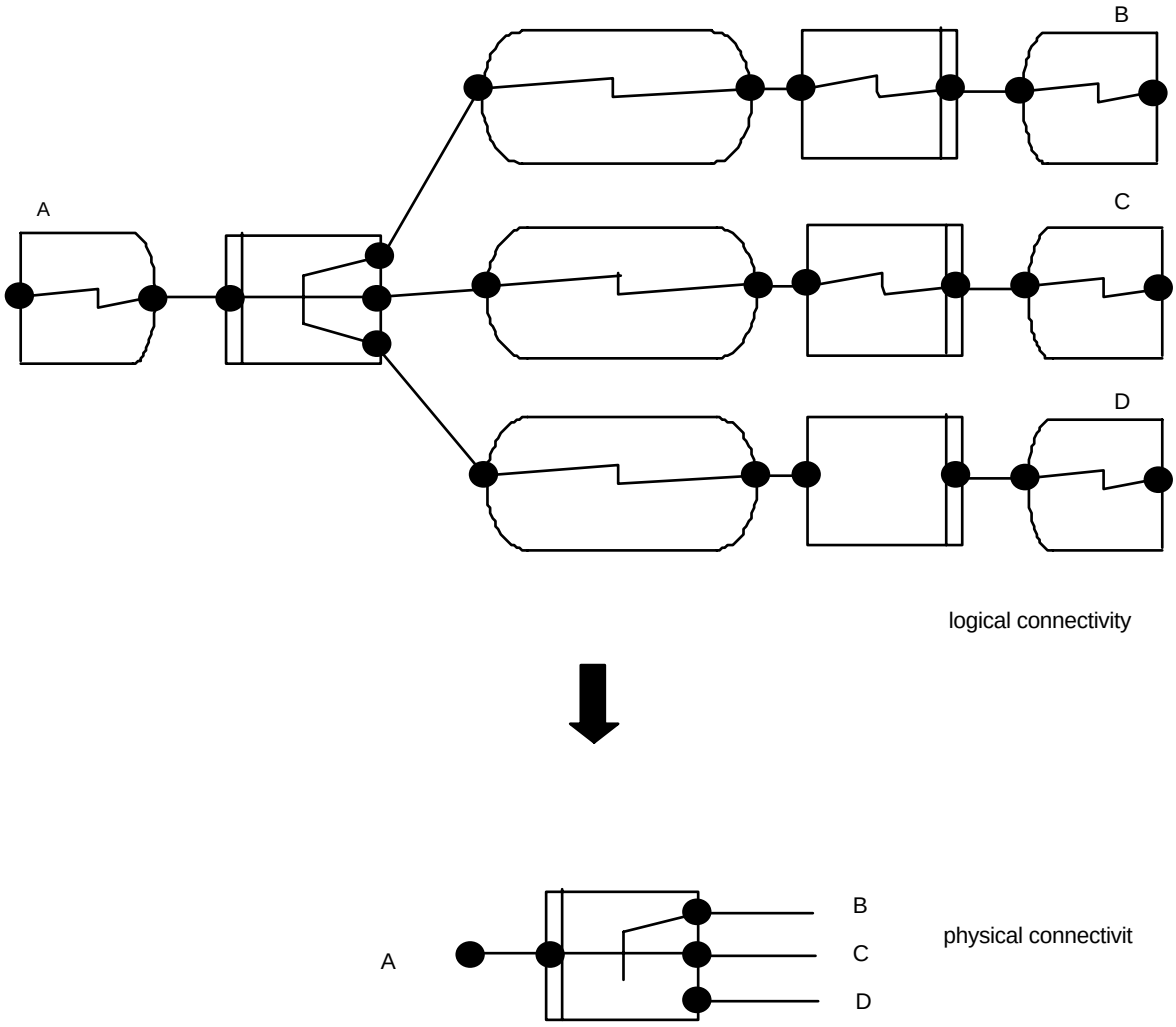


Figure C.7: Example mapping of multiple features to physical connectivity

C.4 Socket behaviour

C.4.1 Requirements

The following requirements and principles should be taken into account when defining the overall behaviour of sockets.

All socket types or sub-types required for IN should be derivable (or specialised) from the general socket class defined below, see Clauses C.5 to C.9 of this annex for details.

The socket should be modelled as an application layer process being defined in such a way that it can be used in different protocol environments: full OSI protocol stack with ACSE, ROSE and possibly RTSE and other ASEs, or SS No. 7 environment with TCAP and supporting protocols. The interface between the application user and the socket should be independent of the protocol environment in which the socket is imbedded. As such it is possible to use the socket based protocols together with other application protocols in order to form complex services. Example of such protocols are:- Mobile Application Part (MAP) for supporting GSM mobile services;

- Mobility Services Application Part (MSAP) (being developed by SPS6-SIG) for supporting all sorts of mobility management;
- protocols for supplementary services (e.g. the protocols being developed for DSS1);
- ISDN Signalling Control Part (ISCP);
- protocols for the Directory (the CCITT X.500 series);
- protocols being developed for OSI Management Information.

C.4.2 Modelling principles

The general modelling principles for the OSI Application Layer are given in ISO IS 9545. However, this standard is now being extended to include object oriented modelling principles. Therefore, a detailed socket model will not be developed until it has been decided by CCITT Recommendation SG XI or ETSI SPS3 whether the current principles or the extended object oriented method should be used for signalling system related protocols. At this stage only an outline of the services provided by the socket to its users will be given since these can be described independently of the application layer structure. The concept of OSI layer services (CCITT Recommendation X.210) allows a "black box" view of the layer service-provider where the interaction between the service-user and the service-provider is described in terms of service primitives.

The socket model at each user can thus be depicted as shown in figure C.8.

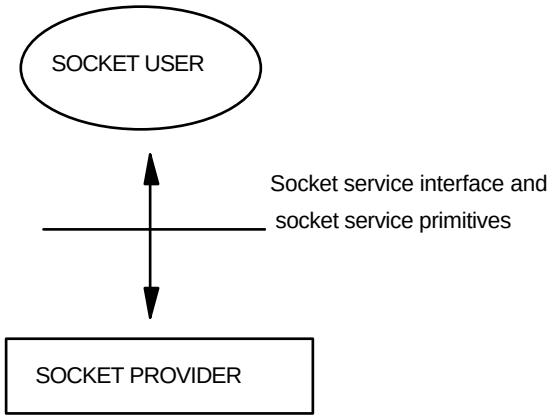


Figure C.8: Socket model

C.4.3 Outline of common socket services

Below only an outline of the common services to be supported by the socket is given. These services are additional to the services defined in Clauses C.5 to C.9 of this annex, which are specific for each socket type (or sub-type).

Detailed specification of the common services and the associated service primitives are for further study.

Socket open

This service is used by the user to request the socket to be established. For some socket types the socket open service is available to both the SSF and the SCF; for other socket types the service is available to only one of them. These conditions are stated for each of the socket types described in Clauses C.5 to C.9 of this annex.

The requester must provide information concerning which socket type (or, for further study, socket sub-type) is to be opened. This information is passed to the peer.

Other information to be exchanged between the socket users, or between a socket user and the socket service provider, is for further study. This may require the use of the Bind operation defined for ROSE (CCITT Recommendation X.219).

Socket close

This service is used by a socket user to request the release of the association established through the socket.

For each socket type Clauses C.5 to C.9 of this annex contain information concerning which party is allowed to close the socket. The socket may be closed by a party other than the one that opened the socket. The socket close service may require the use of the Unbind operations defined for ROSE.

Parameters associated with the socket close service are for further study.

Socket synchronisation

This corresponds to a socket service or a set of socket services that may be required for the following reasons:

- to provide mechanisms for indicating the order in which both common and socket type specific services are to be executed;

- to enable sets of socket services to form single processing activities (e.g. controlling the way in which "atomic operations" between the SSF and SCF can be combined in order to form complex messages);

- to manage synchronisation of the processing, e.g. negotiation of turn and synchronisation points.

The requirements for these and additional capabilities are for further study.

Services equivalent to the above are supported by OSI session layer. They may not be fully supported by SS No. 7.

Socket test

This service can be used by the SSF and the SCF on persistent sockets for checking whether or not the socket still exists and also which logical objects it contains. Additional information concerning the status of the socket and its logical objects may also be exchanged; this is for further study.

Socket user abort

This service enables either user of the socket to release abnormally the association. Reasons for aborting the association could be:

- version not supported;
- user resource limitation;
- user resource unavailable (permanently or temporarily);
- socket type specific problem.

Socket provider abort

This service enables the service provider to report locally that there is a problem with the underlying services and that the association has to be released locally.

C.4.4 Socket type specific services

Socket type specific services are defined for each socket type in Clauses C.5 to C.9 of this annex.

C.4.5 Sequencing of services

The common socket services cannot be used in an arbitrary order. In addition, the sequencing of socket type specific services defined in Clauses C.5 to C.9 of this annex, depends on several elements such as:

- relationship between and synchronisation requirements for the socket specific services;
- type of synchronisation services being available.

The overall rules for sequencing of services cannot be defined until all common services have been defined in terms of service primitives.

C.4.6 Mapping onto other services

The internal structure of the socket (e.g. in terms of Application Service Element (ASE) will be defined when a recommendation concerning modelling of the application layer for signalling has been developed. The internal structure will also determine how services available at the user-socket interface are to be mapped onto the ASEs contained in the socket.

C.5 Connection control socket

C.5.1 Opening and closing of the socket

C.5.1.1 Socket opening

A connection control socket is opened when a feature requiring IN connection control is invoked. It is either opened by the SSF as a result of the detection of a trigger event that satisfies trigger criteria, or is initiated by the SCF, when some service logic indicates to do so.

During the socket opening, information to identify the socket type and the SCF/SSF protocol version understood by the SSF must be exchanged.

socket opened by the SCF:

The SCF can decide to open a connection control socket when some service logic indicates to do so. When opening the socket, the SCF can invoke the CREATE or ATTACH primitive, which enables to create object(s) in the socket.

socket opened by the SSF:

The SSF decides to open a connection control socket when a trigger condition is fulfilled at a given trigger check point (TCP), which corresponds to a state of the Basic Call State Model (BCSM), a phase of the basic call named point in call (PIC), in which an event, named **trigger event**, is received. In other words, when, at a given PIC, a trigger event occurs and the trigger condition is fulfilled, then the SSF decides to open a connection control socket.

Examples of states in the BCSM are: idle, dial waiting, call sent, alert phase, conversation, release phase, ... (FFS)

Examples of trigger events are: off-hook, set-up, alert, answer, busy, facility or release message, wait for answer timeout, ... (FFS)

Examples of trigger conditions are: particular value(s) of called party number, calling line identity or category, release cause, facility element, or some internal SSF information like resource status, service right or call gapping conditions, ... (FFS)

When opening the socket, the SSF can invoke the PROVIDE_INSTRUCTION primitive, which includes parameter(s) to describe the content of the socket, i.e. the objects presented through the socket.

When a trigger condition is fulfilled, the service is not necessarily triggered. Service interaction, as well as filters/sieves set by the traffic management socket must be taken into account (FFS).

The following SDL describes how the SSF handles events received in PIC.

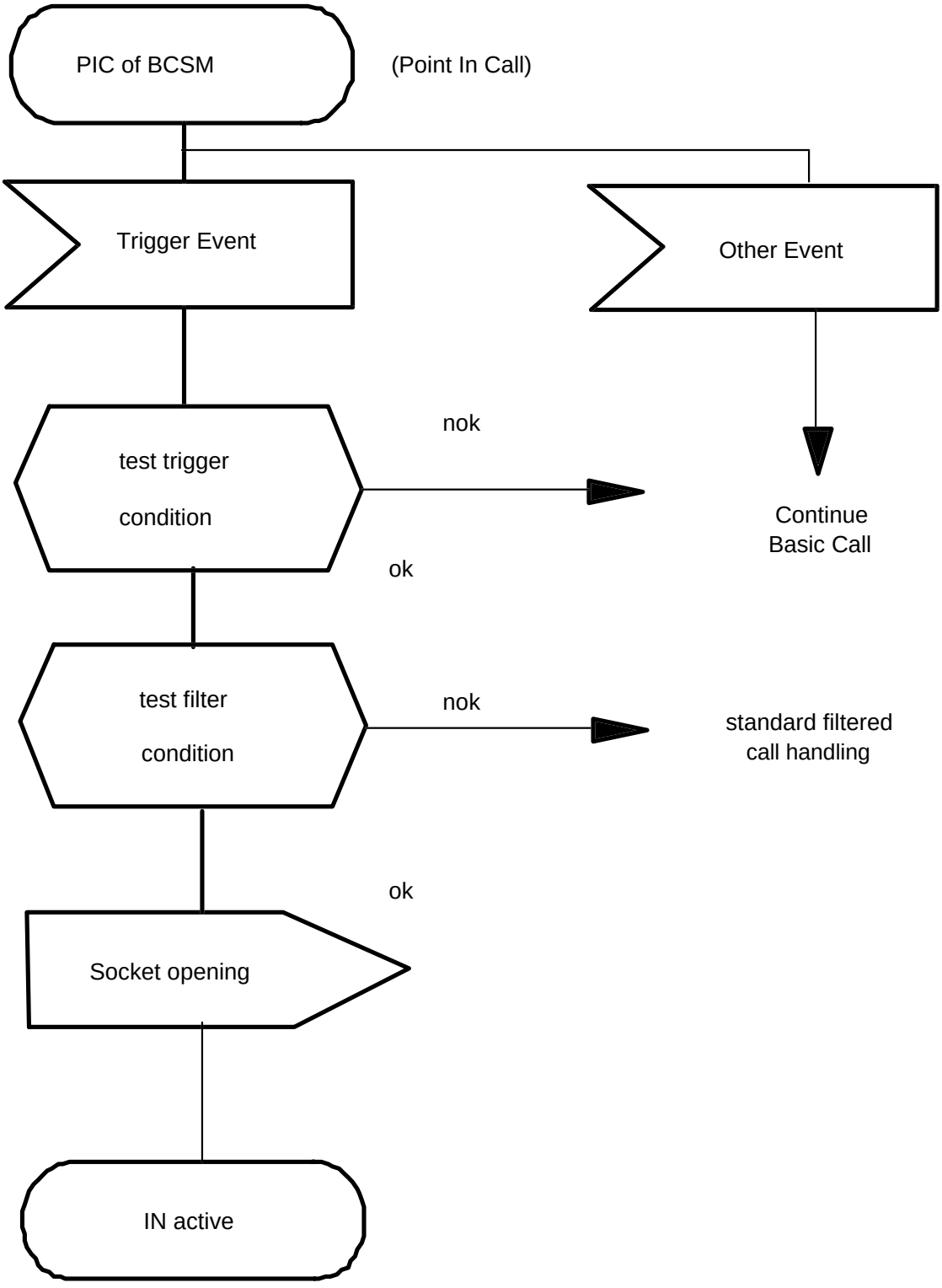

PIC of BCSM
(Point In Call)
Trigger Event
Other Event
test trigger
condition
nok
Continue
Basic Call
ok
test filter
condition
nok
standard filtered
call handling
ok
Socket opening
IN active

Figure C.9

C.5.1.2 Socket closing

A connection control socket is closed when the SCF informs the SSF that the feature is completed and that the socket must be closed.

The SSF may also initiate socket closing, but only in error cases.

C.5.1.3 Socket test

A connection control socket can be tested by the SCF or by the SSF. That means that both SCF and SSF can ask each-other to check whether the socket still exists and also whether a given object still exists within the socket.

Only persistent sockets need to be tested.

The SSF has to test periodically the existence of each socket. The SCF may or may not test, depending on the feature concerned.

NOTE: Primitive(s) enabling socket test are FFS.

C.5.2 Objects in the connection control socket

Two classes of objects are identified in the connection control socket: legs and connection points. The SCF can invoke SSF functions that manipulate these objects, thereby changing their state. Objects in a socket controlled within the context of an SCF-SSF interaction as defined by the socket type. As such, these objects are considered local to the socket.

C.5.2.1 Leg

Definition: A leg is a representation of a communication path towards an addressable network entity, as viewed from the socket (e.g. an end-user, an information sink or source).

A socket may present more than one leg (or even none). Each leg within a socket has a single local identifier.

Each leg has attributes like particular characteristics within the socket ("controlling" or "passive" leg), the charging rate applied to the leg, or the bandwidth associated with the leg.

Legs have states. Operations may modify the leg states. The leg state diagram in subclause C.5.4 shows how primitives modify this state.

Event management on legs: Events may happen on legs in the SSF. To establish communication path towards an addressable network entity, signalling events flow towards and backwards from and towards the direction of this entity. They are processed at the level of the SSF. To allow the execution of services controlled by a service logic at SCF level, it is necessary that the service logic can specify event processing and generation.

Event processing: The SSF monitors each leg for event occurrence as requested by the service logic. The extend of event processing to be performed in the SSF is event and signalling system dependent (e.g. when a release event occurs, the SSF releases the link and sends a release complete message autonomously).

The treatment for the other leg(s) presented in the same socket besides the monitored leg is determined by the service logic with the MONITOR primitive, by means of the following event processing modes:

intercept: this event processing mode results in:

- not processing the event for any other leg of the socket, and
- reporting the occurrence of the event to the service logic.

duplicate: this event processing mode results in:

- processing the event for the other leg(s) of the socket, which were identified in the MONITOR primitive, and
- reporting the occurrence of the event to the service logic.

transparent: this event processing mode results in:

- processing the event for the other leg(s) of the socket, which were identified in the MONITOR primitive, and
- not reporting the occurrence of the event to the service logic.

ignore: this event processing mode results in:

- not processing the event for any other leg of the socket, and
- not reporting the occurrence of the event to the service logic.

Event generation: Associated with the ability to request event monitoring is the complementary ability by the service logic to request the generation of signalling related events on legs. That is done by the SCF with the GENERATE primitive.

List of events that can be: These events must be described in a signalling system independent monitored/generated manner. A minimum set of events includes ALERT, CONNECT, RELEASE.

Default event monitoring: Default event processing modes must be defined. This default modes become active at the moment a leg is created. Values for default event processing modes are FFS, however they should be either intercept or duplicate (otherwise the SCF is "blind"). These default values may depend on the PIC and trigger condition.

C.5.2.2 Connection point

Definition: A connection point is a representation of the interconnection of legs that allows information to flow between legs, only from the perspective of the feature acting through the socket.

A socket may present more than one connection point (or even none). Each connection point within a socket has a single local identifier.

Each connection point may have attributes like the maximal number of legs which can be connected to it.

A connection point has states, representing the number of legs connected to it. So far, no need is identified for primitives creating and releasing explicitly connection points.

C.5.3 Service primitives

Primitives described in this section are as atomic as possible. In other words, none of them is a grouping of several primitives. That allows to describe only once and in detail the atomic interaction/control of the SCF on the SSF. However in the final SSF/SCF protocol, some grouping may be done, for different purposes, including performance issues like minimising the number of "messages" exchanged between SCF and SSF or their volume. Anyway, these grouping should not limit the flexibility that the SCF needs in order to control complex services.

It should be noticed that primitives and procedures are different. Indeed, the procedures include sequencing aspects which are out of the scope of the primitives. This notion of procedure needs further clarification.

NOTE: It may be necessary, in order to manipulate data on a per call basis, to add primitive(s) to manipulate these data in this socket (FFS).

A more detailed description of the primitive will be necessary.

CC_PROVIDE_INSTRUCTION: This primitive is used by the SSF to indicate service activation to the SCF, to request instruction from it.

CC_CREATE a leg: This primitive is used by the service logic to request the creation by the SSF of a leg to an addressable entity.

CC_FREE a leg: This primitive is used by the service logic to request the SSF to release a leg and all associated resources. FREE may be considered the reverse of CREATE.

CC_JOIN a leg: This primitive is used by the service logic to request the SSF to link one or two leg(s) to a connection point and to through-connect the corresponding physical resources (e.g. to through-connect all the legs on a conference bridge).

CC_SPLIT a leg: This primitive is used by the service logic to request the SSF to isolate a leg from a connection point, or all the legs from a connection point, discarding the connection point. SPLIT may be considered the reverse primitive of JOIN.

CC_MONITOR events on a leg: This primitive is used by the service logic 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.

CC_GENERATE an event on a leg: This primitive is used by the SCF to request the SSF to generate on a leg the signalling message corresponding to an event. Information related to this message is given by the SCF in a logical way. It is up to the SSF to translate it in the right signalling system.

CC_EVENT received on a leg: This primitive is used by the SSF to communicate information of one event to the SCF. This event has been previously assigned one of the two modes: intercepted or duplicated.

CC_SEND_RECEIVE information on a leg: This primitive is used by the SCF to request the SSF to send and/or receive specific information to and/or from a call participant. This information may either be inband (e.g. tone, DTMF digits, announcement, ...) or contained in a signalling message (e.g. D-channel message).

CC_RESUME basic call: This primitive is used to allow object control resumption from the SCF to the SSF. With this primitive, the SCF gives back control to the SSF on some objects, like leg(s) or connection point(s). The SSF will then continue to handle these objects with the basic call control.

CC_DETACH a leg: This primitive is used by the service logic to request the SSF to remove a leg from one socket and to assign it an absolute (i.e. single network-wide) identifier, so that it can be transferred to another socket instance, to which the leg was/will be attached by means of the ATTACH primitive using the same absolute identifier.

CC_ATTACH a leg: This primitive is used by the service logic 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 DETACH primitive with the same logical absolute identifier.

CC_MODIFY a leg: This primitive is used by the service logic to request the SSF to modify some characteristics of an existing leg in the current socket (e.g. to change the charging rate applied to the leg, or to change the bandwidth associated with the leg, ...).

CC_JOIN connection points: This primitive is used by the service logic to request the SSF to merge two connection points together. As a result of this primitive, all legs connected to both connection points end up connected to a single connection point.

CC_SEND information on a connection point: This primitive is used by the service logic to request the SSF to broadcast information such as tones or announcements to all legs connected to a connection point.

C.5.4 Leg state representation

The following figure shows how primitives modify the leg state.

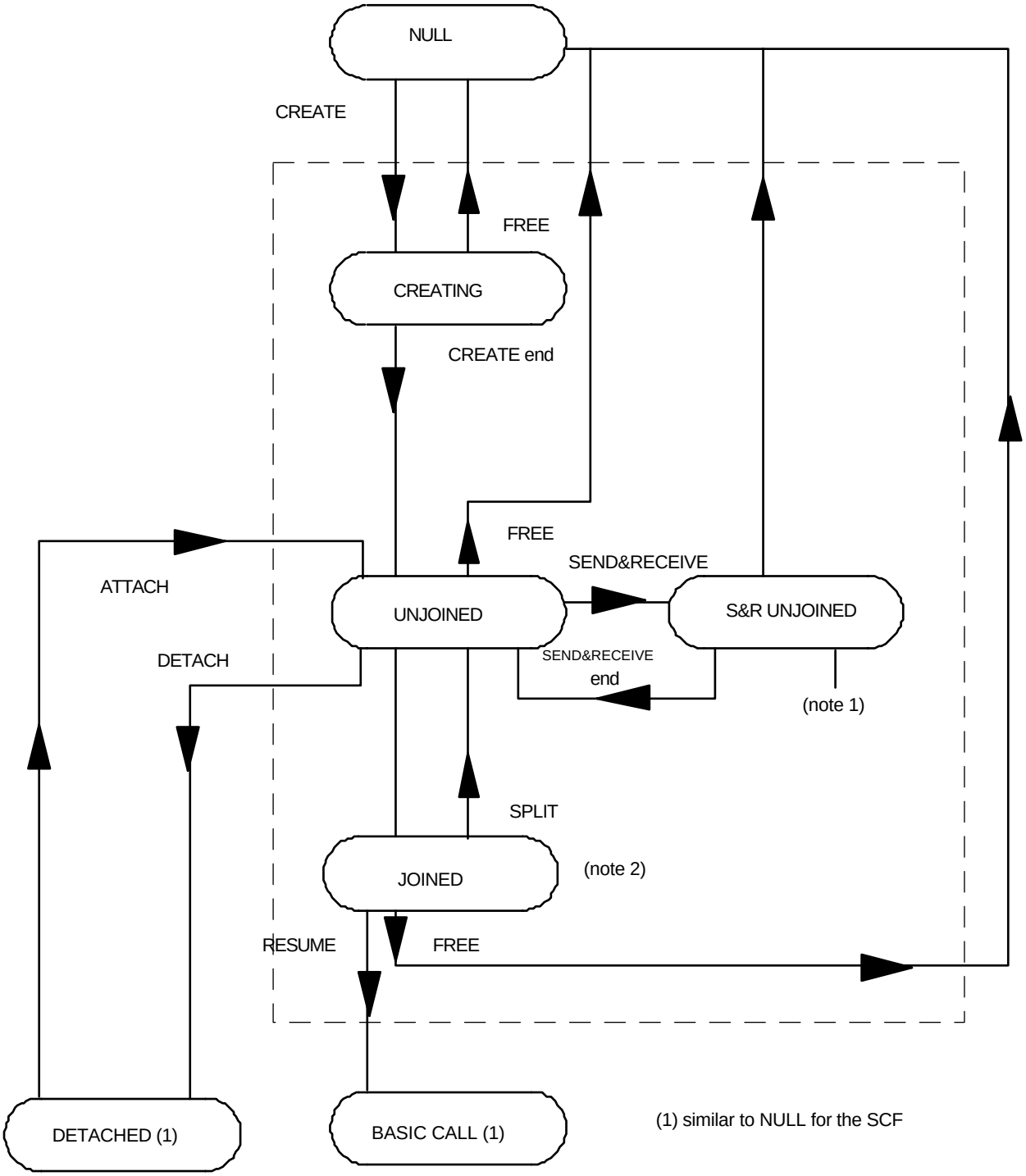


Figure C.10: Leg state diagram

NOTE 1: JOIN in this state waits until SEND&RECEIVE end to be executed.

NOTE 2: SEND on CP does not change the leg state, but the CP state. SEND&RECEIVE on leg must be preceded by SPLIT.

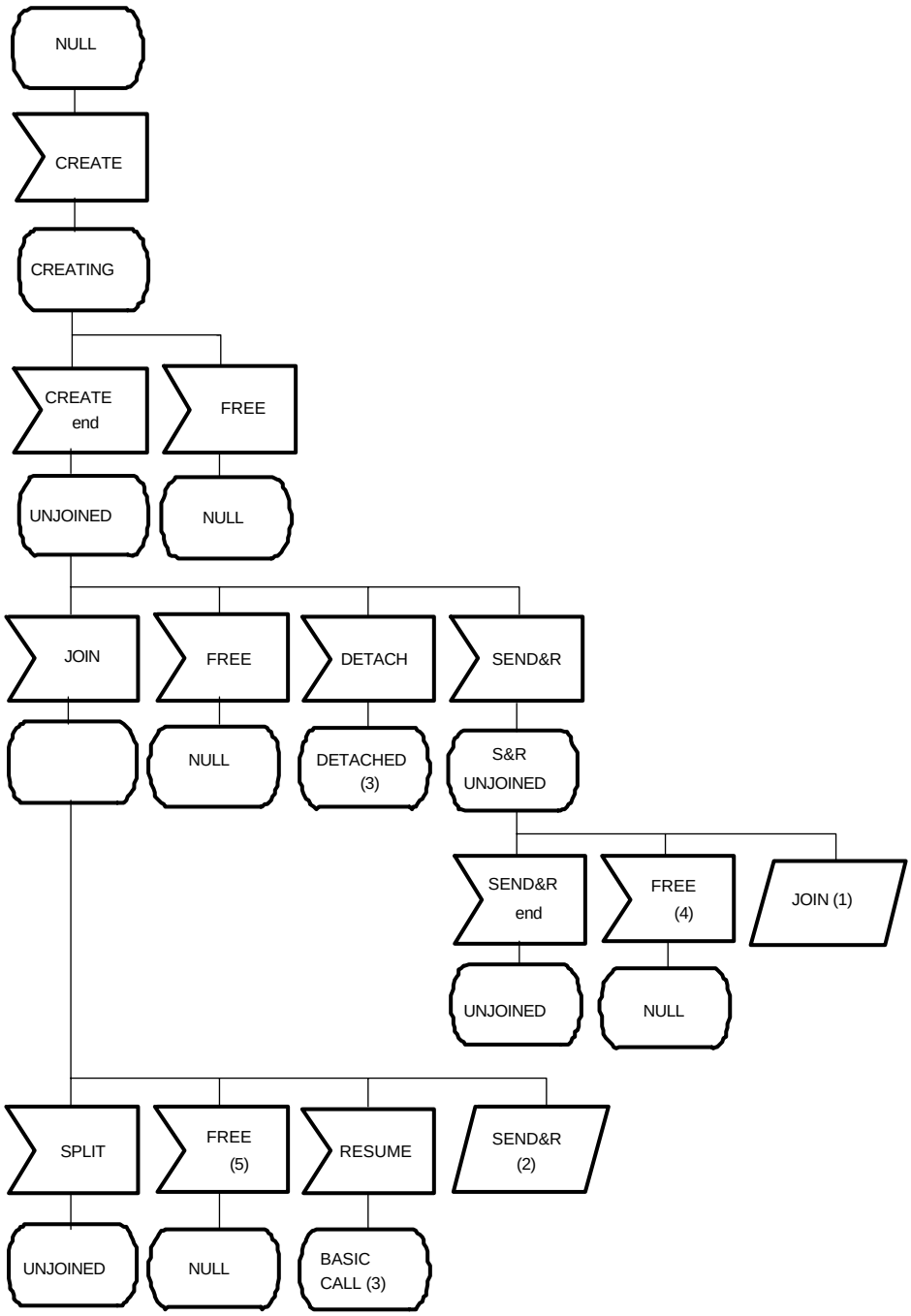


Figure C.11: Leg state transitions in SDL

NOTE 1: JOIN in this state waits until SEND&RECEIVE end to be executed.

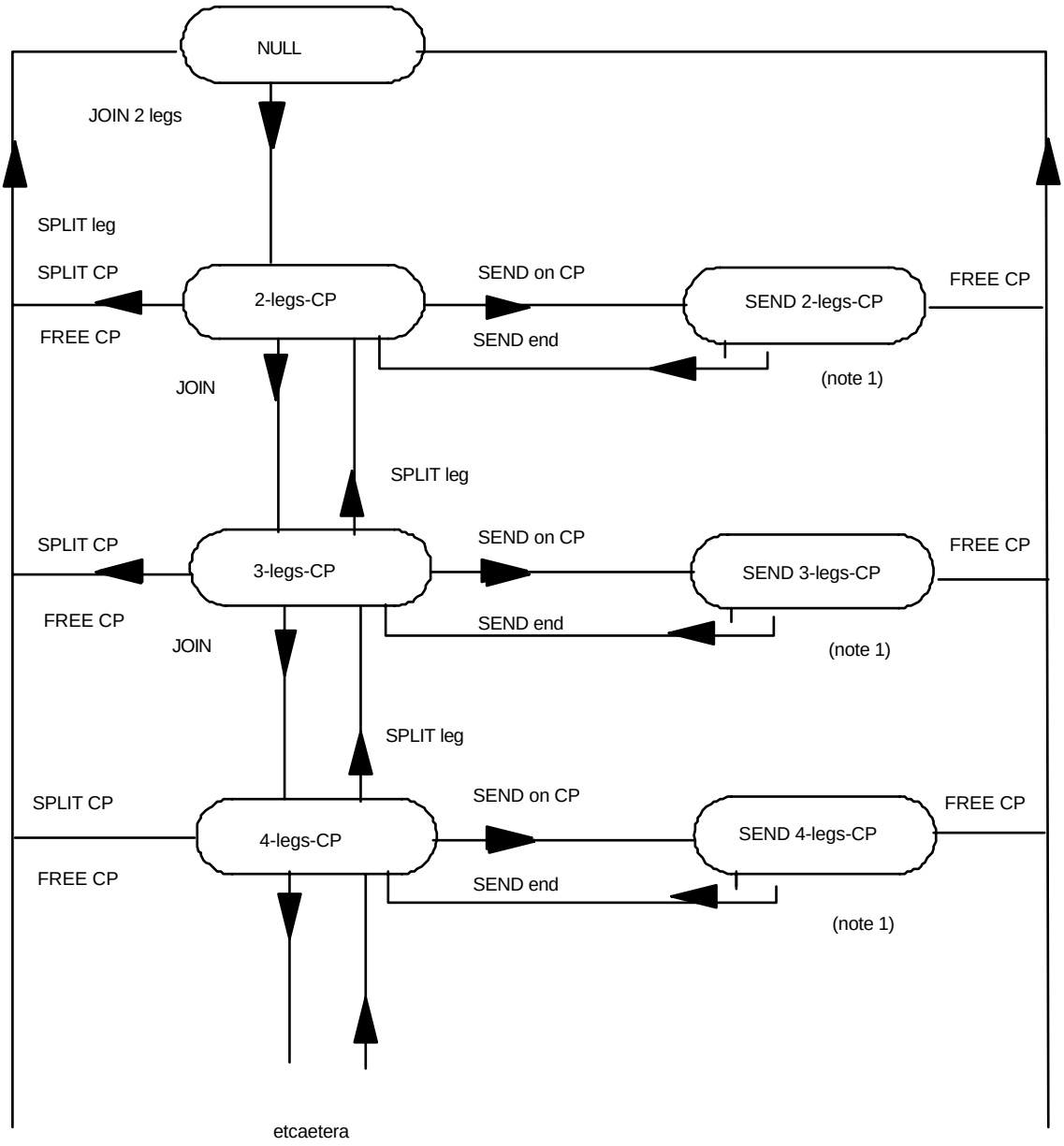
NOTE 2: SEND & RECEIVE on leg must be preceded by SPLIT. SEND on CP does not change the leg state, but the CP state.

NOTE 3: These states are similar to _NULL_ for the SCF.

NOTE 4: Should FREE be possible in this case ?

NOTE 5: FREE the connection point on which the leg is joined.

C.5.5 Connection point state representation



NOTE 1: JOIN or SPLIT in this state waits until SEND end to be executed.

Figure C.12

C.6 Data Management (DM) socket

The data management (DM) socket is used to exchange data between the SCF and the SSF.

C.6.1 Opening and closing the socket

The DM socket can handle two different types of data in the switch:

SIR data (Service Instance Related data): These data are updated on a call per call basis. At least three mechanisms are possible:

Mechanism A: SIR data are manipulated through the connection control socket, then it is necessary to add in the connection control socket the same primitives as those in the DM socket (or a subset of them).

Mechanism B: SIR data are manipulated through a non-persistent DM socket which is opened and closed for each call. This implies to define an absolute reference of the application instance, allowing the connection control socket and the non-persistent DM socket to be related one to each other.

Mechanism C: SIR data are manipulated through a persistent DM socket. This implies that a single DM socket treats many application instances, each of them being referenced within the socket.

SII data (Service Instance Independent data): These data are updated via a persistent DM socket which is service instance independent.

For each of these mechanisms primitives for opening and closing the socket are:

Open (SCF ↔ SSF)

- Parameters: socket_id, ...

Close (SCF ↔ SSF)

- Parameters: socket_id, ...

Both the SCF and the SSF can open and/or close the socket. When the socket is closed, the SSF and SCF has to take care of deleting all processes attached to it.

C.6.2 Objects in the socket

Data accessed through the DM socket are of the following types:

NOTE: The actual model of every data is for further study.

tables of records

This means that primitives in the DM socket will be similar to classical relational database primitives we know in the context of telecommunications, even if the SSF is not supposed to perform the functionalities of a database.

counters

buffers

The actual objects in the DM are the following ones:

trigger table (SII data)

Entries in the trigger table are indexed by couples (Basic_call_state/Trigger_event).

Attributes are: (to be refined in SPS3)

- trigger_parameters,
- trigger condition,
- SCF_address,
- information_to_send ...

subscribers data (SII data)

This includes:

- service_rights,
- service_activation_indicators, ...

Observation data

This includes:

- observation_activation_indicators (SIR data),
- traffic_observation_data (SII data),
- call_observation_data (SIR data),
- message_to_be_edited (SII or SIR data),
- service_specific_counters (SII data, e.g. televote) ...

charging data

SII charging data are TMN matters.

Some SIR charging data are handled through the modify_leg primitive in the connection control socket, but it is not sufficient. For instance, how can the exceeding of a threshold during a call be reported to the SCF ?

NOTE: SIR routing data are handled through the create_leg primitive in the connection control socket, while SII routing data are TMN matters.

C.6.3 Service primitives

For each primitive PR we can define:

- PR.request (PR in abbreviated form)
- PR.result
- PR.error
- PR.reject.

NOTE: The actual class of each primitive is not yet specified in this document.

DM_create (SCF → SSF)

Asks the SSF to create a new data_item (record, counter, buffer). This does not include the creation of a new table but only the creation of a new record in an existing table. The type of data must be specified (record, counter, buffer). If the data item is a counter, the increment must be specified. If the data item is a buffer, its length or its duration must be specified.

Parameters: (to be refined in SPS3)

- data_item_id,
- data_item_information,
- data_item_type,
- counter_increment,
- buffer_length, buffer_duration, data_item_class (static/dynamic) ...

DM_delete (SCF → SSF)

Asks the SSF to delete a data item.

Parameters: (to be refined in SPS3)

- data_item_id ...

DM_modify (SCF → SSF)

Asks the SSF to update an existing data item. If the data item is a counter, the updating can be an incrementation which is specified in the parameter counter_increment, or a reset.

Parameters: (to be refined in SPS3)

- data_item_id,
- data_item_information,
- counter_increment,
- counter_reset_indicator ...

DM_poll (SCF → SSF)

Asks the SSF for the value of given data item, which is to be reported within a poll.result primitive. If the data item is a counter, a parameter indicates whether the counter must be reset after the poll.result.

Parameter: (to be refined in SPS3)

- data_item_id,
- counter_reset_indicator ...

DM_event (SCF ←SSF)

Informs the SCF of the reception of a modify request, that can be a request coming from another IN entity or an end-user, or an updating due to internal reasons. If the data item is a counter, the event primitive can report the exceeding of a threshold which had been specified in a previous report primitive. The event identifier, which has been given in a previous report primitive, must be indicated.

Parameters: (to be refined in SPS3)

- data_item_id,
- data_item_information,
- request,
- request_origin,
- event_id ...

A: If the SSF handles charging, the SCF may need to be informed, in real time, of threshold exceeding.

B: In case of interaction between SSF located features (e.g. basic call forwarding) and an IN service, the updating of a data by the end-user in the SSF (e.g. call forwarding activation) may need to be reported to the SCF.

DM_lock (SCF → SSF)

Prevents any updating of a data (write_lock type) or any reading of a data (poll_lock type) coming from another IN entity.

Parameters: (to be refined in SPS3)

- data_item_id,
- duration,
- unlock_event_list,
- data_request,
- lock_type ...

Examples A and B concerning the report primitive are valid for the lock primitive.

DM_unlock (SCF → SSF)

Totally or partially cancels a previous lock primitive.

Parameters: (to be refined in SPS3)

- data_item_id,
- data_request ...

DM_report (SCF → SSF)

Asks the SSF to report, with a data_event primitive, any updating (or attempt of updating) of a data.

Parameters: (to be refined in SPS3)

- data_item_id,
- duration,
- request_origin,
- event_id, threshold_value ...

DM_no_report (SCF → SSF)

Totally or partially cancels a previous report primitive.

Parameters: (to be refined in SPS3)

- data_item_id,
- request_origin ...

The four primitives DM_lock, DM_unlock, DM_report, DM_no_report can be merged in one primitive DM_monitor, which can replace the four primitives.

DM_monitor (SCF → SSF)

Merges the DM_lock, DM_unlock, DM_report and DM_no_report primitives in one single primitive.

Parameters: (to be refined in SPS3)

- data_item_id,
- data_request,
- duration,
- request_origin,
- monitor_mode,
- values of monitor_mode:
  1) lock & no_report (similar to discard)
  2) lock & report (similar to intercept)
  3) unlock & no_report (similar to transparent)
  4) unlock & report (similar to duplicate).

DM_start_collection (SCF → SSF)	Instructs the SSF to start counting some event. It only address counters. Parameters: (to be refined in SPS3) - data_item_id, - counter_reset_indicator ...
DM_cancel_collection (SCF → SSF)	Cancels a previous DM_start primitive. Parameters: (to be refined in SPS3) - data_item_id, - counter_reset_indicator ...

C.7 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 the 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 operations may be performed on the supervised objects via other types of sockets with access to the supervised objects.

The execution of these independent operations 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.

C.7.1 Opening and closing of the socket

The status monitoring socket may be opened and closed by the SCF in the following ways:

- Opening and closing for the status monitoring of a single object identity.
  - To poll the current state.
  - To request the next state change.
  - To request the future state changes.
- Opening and closing for the status monitoring of multiple object identities.

The requests for monitoring are spread in time and there is no direct relationship between the different requests. The requests to poll for the current state, or the next state change(s) may be mixed for different object identities.

In this case reference information is to be exchanged to link the replies on the previously invoked requests. It is up to the SCF to decide when the socket is to be closed. This case can be considered as a permanently opened socket.

C.7.2 Objects in the socket

The status monitoring socket contains these objects wherefore the SCF can request busy/free supervision during the execution of service logic.

A list of possible objects is the following:

- addressable entities;
- queues;
- resources for communication with addressable entities.

C.7.3 Service primitives related to the objects

The following service primitives shall be provided:

SM-Start Status Monitoring The SCF application uses this service primitive to request the monitoring of busy/free status changes of a specified object.

NOTE: 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 for further study.

SM-Cancel Status Monitoring The SCF application uses this service primitive to request the cancellation of a previously requested monitoring.

SM-Poll Status The SCF application uses this service primitive to request the busy/free state of the specified object.

SM-Event Status This service primitive is used to report a status change on an object, previously placed under supervision by the SCF application.

C.8 Traffic management socket

NOTE: At this point in time, only traffic *limitation* mechanisms are considered; other traffic management mechanisms such as dynamic traffic control procedures (e.g. re-routing) are for further study.

The IN Traffic Management (IN TM) mechanisms limit the number of calls that are allowed through an IN-structured network, by filtering calls with given characteristics. The filtering is applied only to those calls related to an IN-provided service, that request the assistance of IN functions in order to be executed: they are termed IN calls.

Calls that do not request the assistance of IN functions ("non-IN calls") are not affected by IN Traffic Management mechanisms.

The interactions between SSF and SCF for the provisioning of such mechanisms in an Intelligent Network are modelled by the Traffic Management (TM-)Socket. This is a non call-related socket.

The functionalities needed in the SSF in order to support this socket type and all the objects and primitives implied in it, are specified in SPS3.

The objectives for introducing IN Traffic management mechanisms into a telecommunications network, as well as the rationale behind the choices made for the model, are contained in the TR of SPS3. They are not analysed here as they are transparent to the socket model, that is, they are not reflected in a specific object or primitive, but rather underlie the whole model.

C.8.1 Opening and closing of the socket

IN 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.

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 SSF) is not ruled out, and needs further study.

The execution activity implies an interaction between the SCF and the Service Switching Function (SSF), which is modelled by primitives on objects in the Traffic Management Socket (TM-CANCEL, TM-CREATE, TM-EVENT, TM-MODIFY, TM-POLL).

A TM-socket will be **opened** between a given SSF and a given SCF at the time of activation of the first TM-filter by that SCF into that SSF. The socket will remain open as long as at least one filter is active in the SSF, which was previously activated by that SCF.

Therefore, a socket is always opened by the SCF and the first primitive is a TM-CREATE, while the socket may be closed either by the SCF, as a result of a TM-CANCEL primitive on the last active filter, or by the SSF as a result of the expiration of the last active filter. In the latter case, a TM-EVENT primitive may be issued by the SSF prior to the socket closing.

C.8.2 Objects in the socket

The traffic management socket contains 1 type of object: the IN TM filter. There may be one or several filter instances within the same socket. Each of them has the following attributes:

[1] filter scope,

[2] filter type,

[3] filter severity,

[4] filter duration,

[5] call treatment.

The object attributes are used as parameters in the primitives, and they are detailed in the TR or SPS3. The first two parameters together, scope and type, form an identifier of the filter which is unique within one SSF.

A filter object in the socket corresponds to an IN TM "physical" filter which is activated in the SSF, after an activation order is received from the SCF (see NOTE).

C.8.3 Primitives on the sockets

The following is a list of the primitives on the IN TM filter object, and a brief description of each of them. The details of the primitives are described in the TR of SPS3.

CANCEL IN TM filter(s) The TM-CANCEL primitive is used by the SCF to request the immediate deletion of one or more logical filter object(s) in the TM socket (i.e. the immediate de-activation of one or more filter(s) in the SSF).

CREATE an IN TM filter The TM-CREATE primitive is used by the SCF to request the creation of a logical filter object in the TM socket (i.e. the activation of a filter in the SSF).

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

MODIFY an IN TM filter The TM-MODIFY primitive is used by the SCF to request the modification of the attributes of an existing logical filter object in the TM socket (i.e. the modification of an already active filter in the SSF).

POLL IN TM filter(s) The TM-POLL primitive is used by the SCF to request status of the logical filter object(s) in the TM socket (i.e. the number and the type of filter(s) that are active at that time in the SSF as a consequence of activation orders coming from the SCF).

NOTE: Since a SSF may interact with several SCF's at the same time, it is possible that two different SCF's may request an operation on the same active "physical" filter: this means that there may be two objects in different sockets corresponding to the same physical entity. However, these collisions are typical of shared resources, and they should be handled entirely within the SSF, i.e. they are totally transparent to the interface being modelled with the TM socket.

C.9 Resource control socket

This socket type is introduced in the connection control model to allow the SCF to operate on resources (objects) used for information transfer between addressable entities and the network during the execution of service logic. Information 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 physical allocation of the resources are known in the SCF determines the resource control.

Following possibilities shall be considered:

a) 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 as 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.

b) 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 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 service logic.

C.9.1 Opening and closing of the socket

The resource control socket may be opened and closed in the following ways:

- for a single resource control request
- for multiple resource control requests (e.g. permanently open socket)

It is up to the SCF to decide when the socket will be closed, the SRF shall support both cases.

C.9.2 Objects in the socket

The following objects may be addressed via the resource control socket:

- DTMF receiver
- Tone recognition devices
- Voice recorders
- Message receivers (e.g. DSS1 defined message elements)
- Announcement sender (fixed, variable)
  - speech
  - message (e.g. DSS1)
- Tone generator
- Synthesised voice/speech recognition devices
- Protocol converters
- Interactive prompting facilities
- Conference bridges
- Other objects are for further study

C.9.3 Service primitives related to the objects

The following service primitives shall be supported:

RC-RESERVE RESOURCE

The SCF uses this service primitive to request the reservation of a resource. This is to guarantee that when the RC-RESERVE RESOURCE primitive is executed, the resource is available. A time reference indicates when and for how long the SCF wants to reserve the resource. As long as the referenced time is not reached, the resource may be reserved or allocated to other service instances.

The reply on this primitive shall be a "resource reference number" that is to be used in the RC-SEND&RECEIVE service primitive.

The use of this primitive and the possibility of combining it with other primitives are for further study.

RC-ALLOCATE RESOURCE

The SCF uses this service primitive to request the allocation of a resource.

For the allocation of announcement resources it shall be possible to identify standard announcements, service specific announcements and user specific announcements.

The reply of this operation shall be a "resource reference number" that can be used by the public network to route a call to the SRF (i.e. to the requested resource). This resource reference number is further used by the SCF to identify the resource in the further primitives.

RC-RELEASE RESOURCE

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

RC-SEND & RECEIVE

The SCF uses this service primitive to request the sending/reception of information via the specified resource. This information may either be inband (e.g. tone, DTMF digits, announcement) or out of band (e.g. D-channel message).

The associated parameters contain following info:

- resource reference number.
- specification of send portion, with possible send stop options.
- specification of receive portion, with possible receive stop options.

The service primitive consists of:

- a send sequence.
- a receive sequence.
- a response sequence.

The send sequence, the receive sequence and the response sequence may or may not exist, depending on the parameters.

Information is sent to the addressable entity during the send sequence in which the possibility exists that the receive sequence is activated with/without terminating the send sequence, as indicated in the service primitive parameters.

The receive sequence terminates as indicated in the parameters (e.g. T/O, number of characters, stop character).

The response sequence can be activated during the receive process or after the receive process.

This service primitive is terminated when the requested function is executed or after an error reply is returned to the SCF.

C.10 Requirements on the basic call model

In this clause only the requirements on the basic call model are treated, as the specifications of the model are worked out in detail by SPS3, based on these requirements. In the TR of SPS3 an example of the Basic Call Model is included.

C.10.1 The basic call model versus the connection control model

In an IN-structured network the SCF has to deal with calls which were controlled by basic call control functionality until some trigger was generated and IN Service Control was invoked. As IN Service Control makes use of the (network independent) Connection Control Model (CCM), there must be a way of relating the state of a basic call with its representation in the CCM.

For this purpose the Basic Call Model (BCM) is needed, which describes the basic call and allows a translation of its state into the Connection Control Model once a service is triggered and the SCF takes over call control. This model must contain all information the SCF needs to know about a call, immediately after triggering.

As there does not exist a unified description of a basic call, it will be necessary to map known, network dependent, BCM's onto the CCM (and v.v.).

In figure C.13 a possible view of the relationship between the SCF, the SSF, the CCF (Connection Control Functionality) and the BCF (Basic Call Functionality) is shown.

The SCF communicates only with the SSF. A part of the SSF handles Connection Control, which is named CCF. The basic call functionality can be seen as a subset of the CCF, as only a subset of its capabilities are needed for a basic call. (Whereas the CCF contains all connection handling capabilities of the SSF (as viewed by the SCF)).

However, this does not mean that the BCF is active at the same time as the CCF. When a service is activated, the SCF takes over control of the call by controlling it via its interface to the SSF. The BCF is not active in this situation.

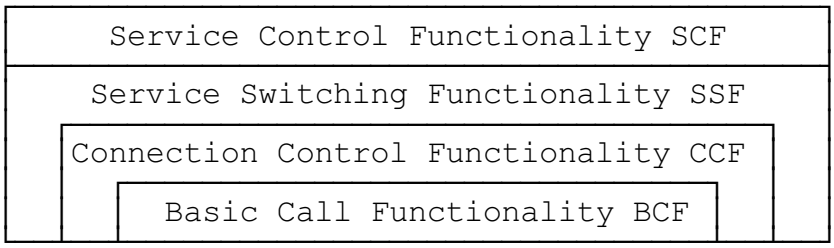


Figure C.13: Relationship between SCF, SSF, CCF, and BCF

C.10.2 Call state transition requirements

Within an IN-structured network there is a need to describe the basic call by means of a model. From this model the SSF obtains information about the state of an existing call when a service is triggered and is able to present the actual connection and its state to the SCF by means of the CCM. This does not mean the SCF needs to "know" everything about a basic call, only that information necessary to perform its function as Service Control.

Beside this it is not allowed to the SCF to hand back control over a call to the SSF at any time. This can only be done if the Basic Call Functionality of the underlying network is able to further handle the existing call autonomously. In this situation the Basic Call Model can be used as well, to determine whether or not the BCF is able to take over call control.

To achieve this, the basic call can be described by a Finite State Machine (FSM, see CCITT Recommendation Z.100), which identifies the states and possible transitions of a (basic) call which are of interest to the SCF or the SSF. When describing a basic call, trigger check points may be identified, which indicate the points in Basic Call Processing at which triggering conditions can be checked and service execution can start.

A part of this BCM will be the handling of a TCP (Trigger Check Point). This can be described in figure C.14.

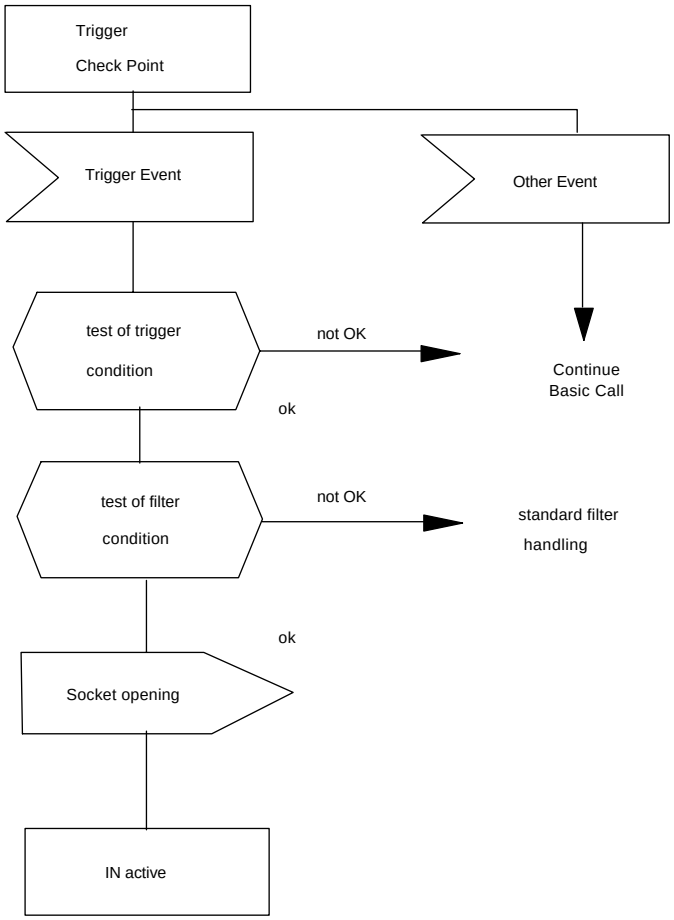


Figure C.14: Handling of events in a Trigger Check Point

Annex D (informative): Examples of call configuration

D.1 Originating services during call set up

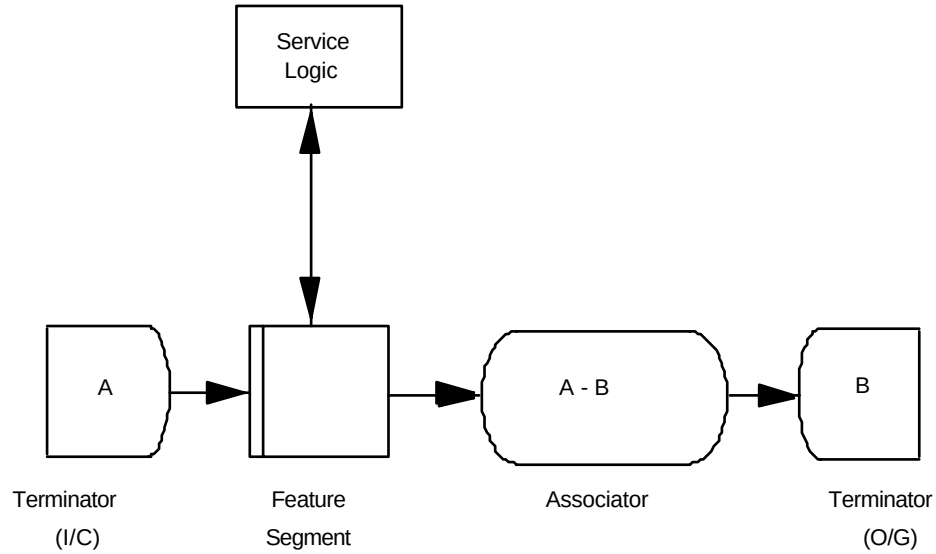


Figure D.1: Originating services during call set up

Typical examples of originating IN services which influence the set up phase of a call are speed calling, 800 service, and outgoing call screening. Most are non-persistent, that is, the feature segment employs a single request/response relationship with the service logic (e.g. for a number translation) during which the call processing in the SSF is suspended. There may be a "sequence" of IN services of this type each requiring a separate feature segment instance to break-in to the segment chain, e.g. speed calling number, which translates to an 800 number, which translates to a DN.

D.2 Terminating services during call set up

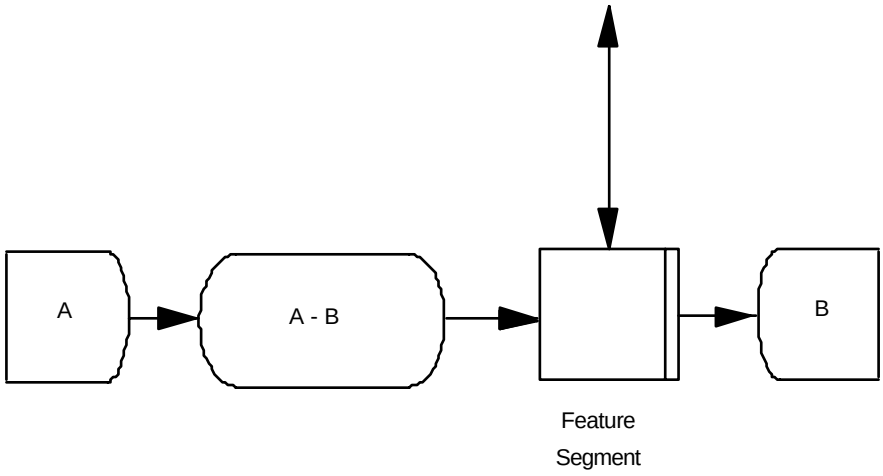


Figure D.2: Terminating Services during Call Set Up

Terminating services during call set up can be either non-persistent (e.g. I/C Call Screening) or persistent (e.g. Call Diversion). They can be triggered on the terminating user (or trunk group) identity, its status (e.g. busy) or an event relating to the presentation of the call (e.g. time out on answer). These trigger conditions are detected by the O/G terminator.

In the case of call diversion, the call is diverted to another DN using a new associator instance, as shown in figure D.3 below. User B terminator is released.

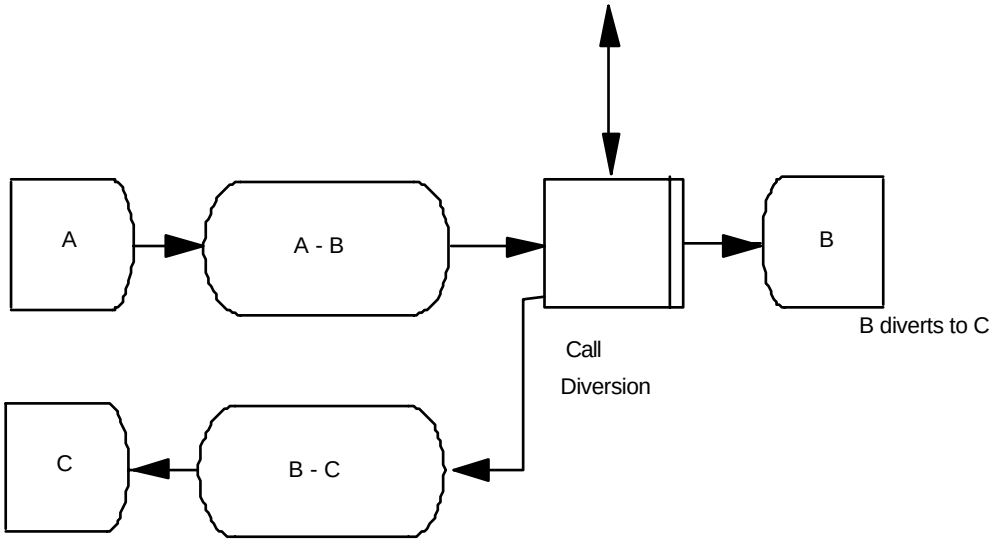


Figure D.3: Terminating services during call set up

D.3 Creating a Leg to an IP

Any service can create a leg to an IP, if necessary, in order to collect a PIN, for example. This is achieved via an associator instance in the normal way. Subsequent to the release of the leg to the IP, the call may be routed to another DN using a new associator instance.

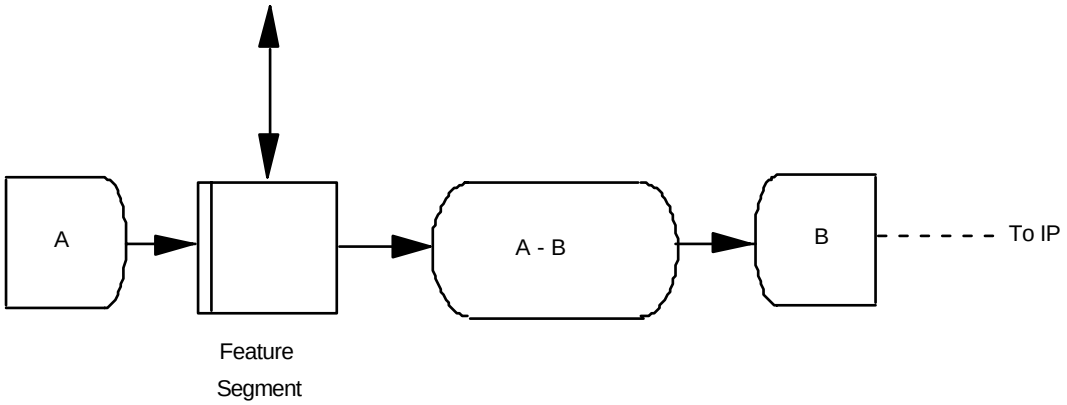


Figure D.4: Creating a leg to an mP

D.4 Multi-party call (mid call trigger)

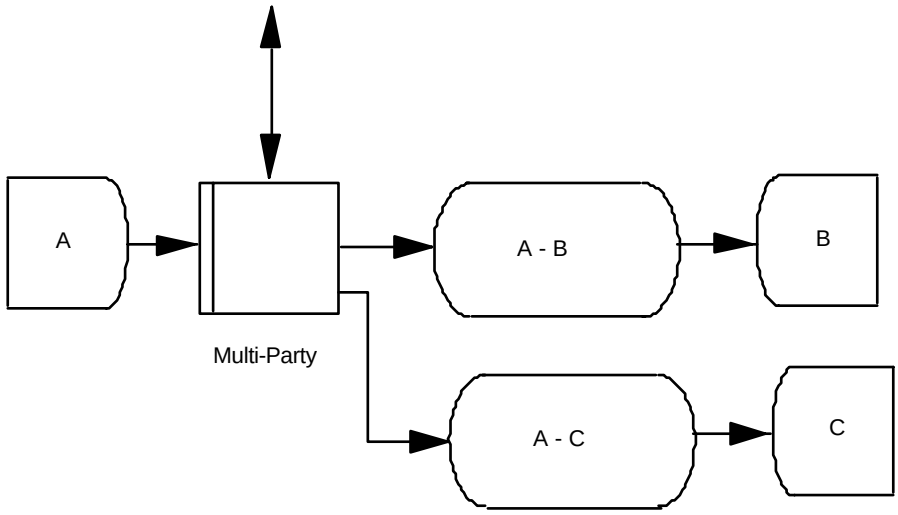


Figure D.5: Multi-party call

Recall (or hookflash) on the original basic call (A-B) is detected in the A-B associator, causing a trigger and the break-in of the Multi-party feature segment into the segment chain. The originating user then dials C, causing the creation of the enquiry leg A-C.

If C subsequently invokes a second multi-party call, the configuration is as shown below.

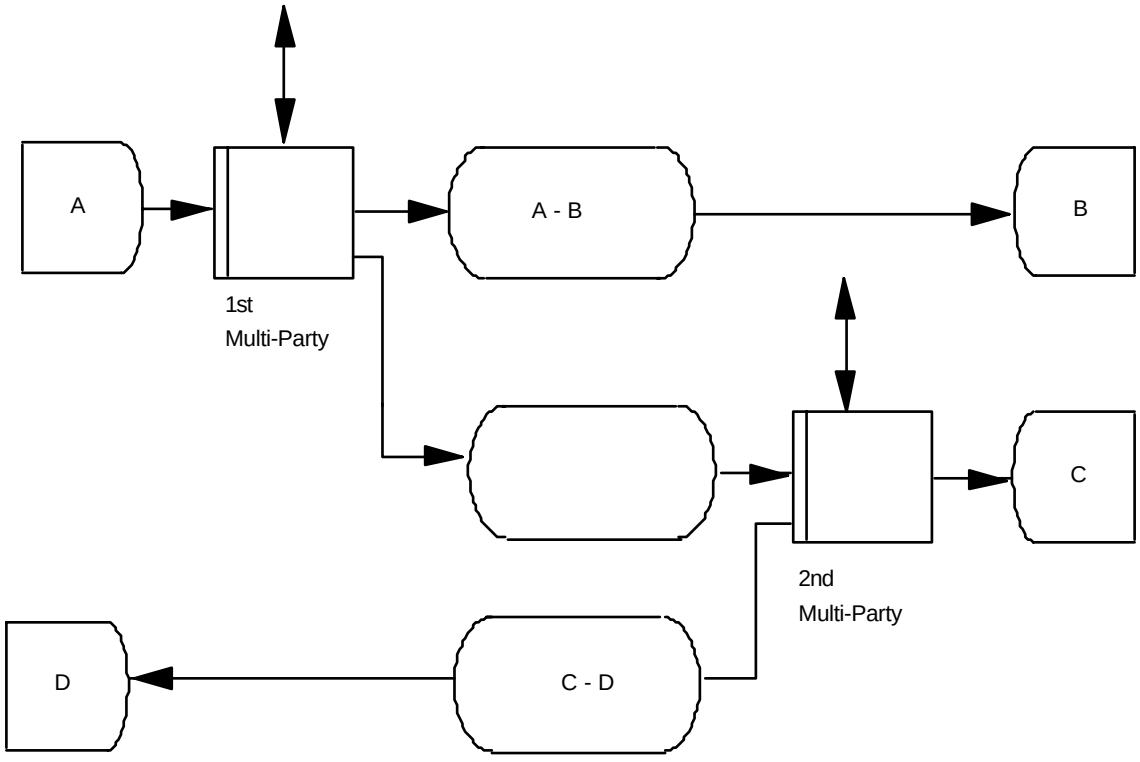


Figure D.6: Multi-party call

D.5 Call Waiting (Barge-In Trigger)

Call Waiting is triggered by the arrival of a new terminating call (from C) for a user (B) already busy on an existing (A-B) call. The existing call may be interrupted, and re-configured, if the called user chooses to accept the call. This service may also incorporate other options, such as voice messaging (via IP) or diversion of the call attempting to barge-in. Consequently more than one connection point may be required within the feature segment.

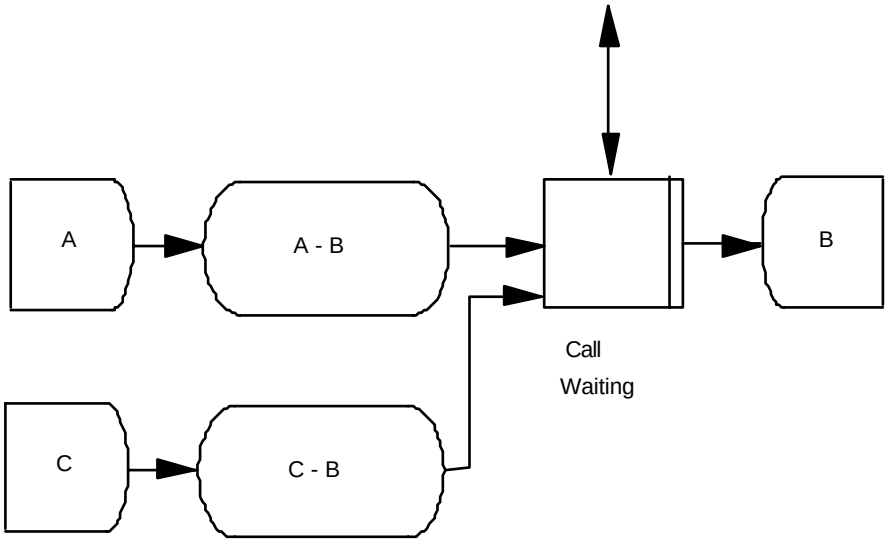


Figure D.7: Call waiting

D.6 "Monitoring" services

These services essentially monitor a call or an individual association within a call, e.g. for time duration or charge incurred. If originating user is being monitored, independent of the services which he may choose to invoke, then the corresponding feature segment is positioned adjacent to the terminator. If a particular association is being monitored (e.g. to a specific destination) then the corresponding feature segment is positioned on the incoming side of the particular associator.

The example below shows a 3 party call in which the originating user is being monitored (e.g. to check that his charge limit is not exceeded), and an individual association (e.g. a leg to an international number) is being monitored to check for a duration limit.

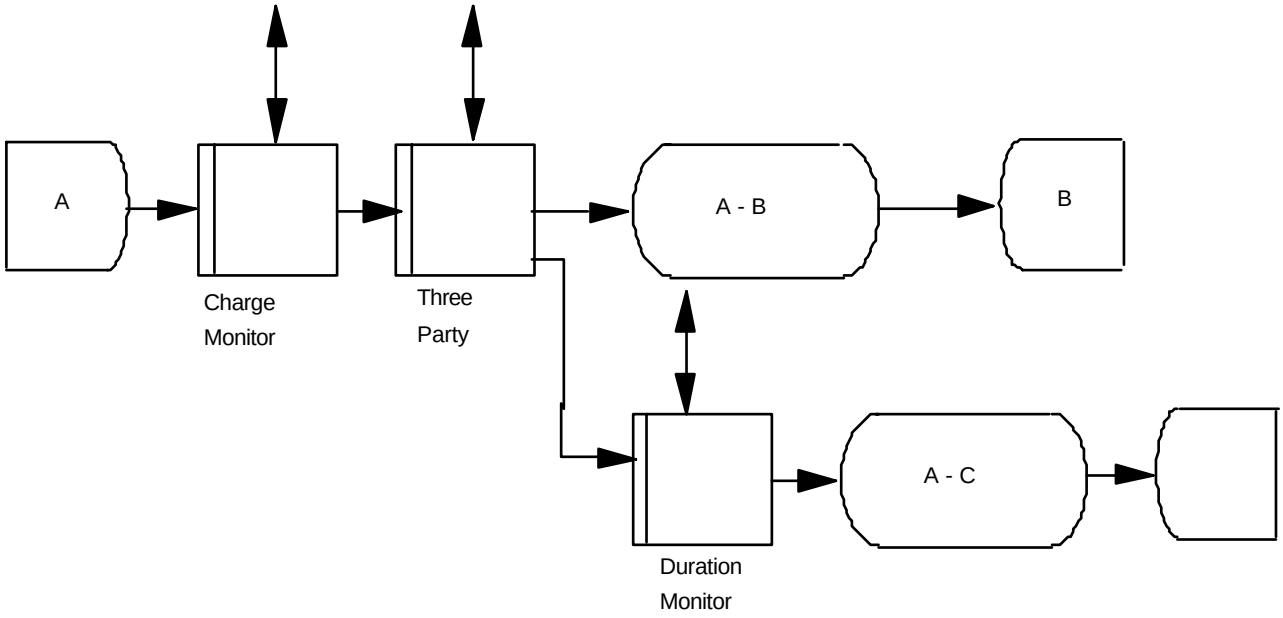


Figure D.8: "Monitoring" services

D.7 Multiple simultaneous services

Clearly there are many possible scenarios. The example below shows an originating user who is being charge monitored, who invokes a 3 party service, using an 800 number on the enquiry leg. The destination user has incoming call screening active, and call diversion on busy.

Note that since the I/C screening service is non-persistent, the corresponding feature segment will have broken-out of the chain prior to the call diversion segment being invoked.

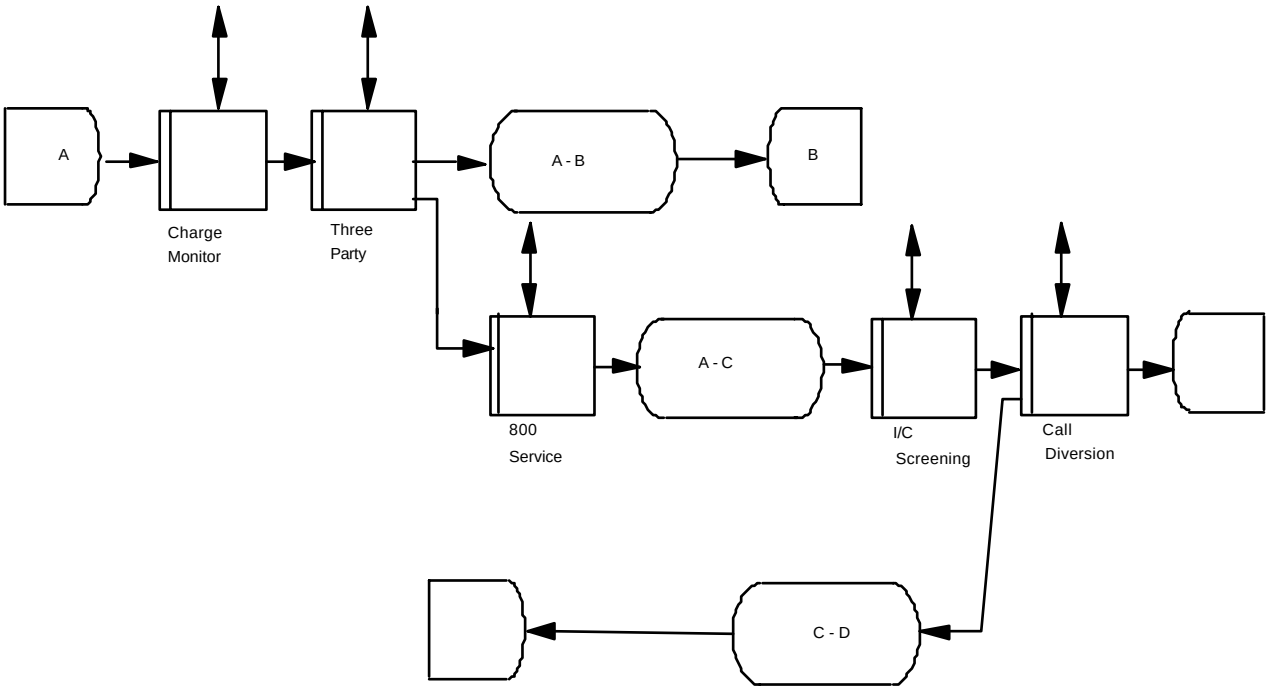


Figure D.9: Multiple simultaneous services

History

Document history	
June 1991	First Edition
March 1996	Converted into Adobe Acrobat Portable Document Format (PDF)