To Conrado
and our families in Yugoslavia and Cuba

The Internet Integrated Intelligent Network

by

Jelena Vasić, Dipl.Ing.

Thesis submitted as a requirement
for the attainment of the degree of
Master of Engineering

Supervised by Dr. Thomas Curran

School of Electronic Engineering, Dublin City University

September 1999

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Jelena Vasić **ID No.:** 93 701 209

Date: 24/09/99

Table of Contents

The Internet Integrated Intelligent Network

M.Eng. Thesis by Jelena Vasić

Abstract

This thesis is based on research, design and prototype implementation work done towards defining an Intelligent Network system integrated with Internet functionality and potential services to be provided by such a system. The main driving force behind the project has been an idea to explore the possibility of adding value to services offered by the two most ubiquitous public communication networks of today, by connecting those networks. The goal of the project was to define such a connection and the services that would become possible through its existence.

The thesis begins with presenting the background to the project. First of all, it contains the definition and description of Intelligent Networks, as the standard means of providing, in telephone networks, services other than basic connectivity. Further, it defines the Internet and looks at the already existent computer network – telephone network integration systems.

In the second part of the thesis, the design of the Internet-Integrated Intelligent Network (IIIN) system is laid out. The IIIN has been defined using definition methods already standard for the IN, as far as this was possible with regard to the entirely different nature of the Internet. The fundamental role of the World Wide Web Internet application in the functioning of the designed system is shown. Following the research and design work description is an account of the implementation work done to build an IIIN prototype and a group of sample services chosen and implemented to demonstrate IIIN functionality. This account includes a list of demonstration scenarios that have successfully been tested on the system prototype.

Table of Figures

Acknowledgements

First of all, I would like to thank Dr. Thomas Curran for supervising and supporting me throughout the work on this project. I would also like to thank Dr. John Nelson for carefully reading the text of this thesis and giving me useful comments. I thank all the people in room JG17 for helping me at every step in the learning process that the work on this project was and for contributing to a very pleasant working environment. I thank my family and friends, who gave me constant moral support, and last, but certainly not least, Conrado, who encouraged me to the end.

1 Introduction

1.1 Overview

The development of information technology (IT) and the emergence of software originally brought dramatic changes to telecommunications [Benn93]. Software allowed for easier and faster deployment of sophisticated systems, which also became more easily manageable and configurable. The first examples of such development, computerised switches with "fancy" features, such as call forwarding and speed dialling, appeared in the 1960s [Fayn97].

The "softwarisation" of telecommunications took a further step with the idea of utilising common channel signalling (designed for the purposes of long distance routing) in order to physically separate special feature data processing from the switching entities in the network. It is through this effort, towards the centralisation and reuse of data processing resources and towards the separation of equipment production from the definition of ways for its utilisation, that the Intelligent Network was born.

The Intelligent Network (IN), a specific telecommunication system invaded and improved by IT phenomena, is today the standard platform for the development and deployment of services other than basic connectivity, which are becoming a significant factor in the growth of telecommunication markets today [Ovum93].

On the other hand, as within the developments in the IT world the accent migrates from concentrated power to networking and connectivity, it seems a logical progression that such systems of computers interconnected in their own specific way (and why not mention the Internet?) should find their way into telecommunication systems. In fact, conglomerations of computer network and telephony functionality have already existed for some time in the form of Computer Telephony (CT), defined and described in [Stra96], which is enterprise-oriented and operates in the world of Local Area Networks (LANs) and PBXs. The booming of the Internet has prompted extensive research and development in the area of integrating computer network and telephone network functionality, on a larger scale than that of CT. The specific case of interest here is the integration of the Internet and the Intelligent Network.

The question remains – what for? First of all, the Internet can contribute to the data processing functionality of IN services with its wealth and variety of information and processing resources. Further, the Internet should allow for location-independence: a) of service-relevant data, which could then be maintained and configured locally by the service provider or user and b) of service deployment and configuration activities, also accessible to a wide group of service providers and users through the Internet. Finally, owing to the universal presence of the Internet, increasing numbers of IN end users are becoming potential users of future IN services which could involve real-time communication by the user on both the Internet (through a computer, most likely a PC) and the telephone network (through a telephone set). Two flavours of such services are possible: services that involve communication on the Internet solely as a means of control (as in the Click-to-Dial service, where the user "orders", through the computer network, that a call be made in the telephone network) [PINT99] and those that involve communication on the Internet for the transmission of content (as in the dramatically developing Internet Telephony) [Low96][IEC99].

The type of integration of the Internet and the IN considered in this master's project is that which enables such services as may involve dual real-time participation of the end user, as an end user both of the IN and of the Internet. A general platform for the provision of such services has been designed as an extension to the Intelligent Network, based on the existing Intelligent Network standards. This platform, which simultaneously includes Intelligent Network and Internet elements, has been named the Internet Integrated Intelligent Network (IIIN).

Intelligent Network services are designed to be built out of service-independent chunks of functionality, called Service-Independent Building Blocks (SIBs), defined in [Q1201] and the current standards define the set of SIBs needed for the building of currently provided services, which are described in [Q1213] and [Q1223]. In the project, several services have been identified and described for the IIIN, as well as a group of SIBs necessary to build those services.

Minimal interfaces for application-level communication between all the actors in the designed system have been defined, both in IDL (defined in [OMG95]) and using a simple text-based protocol similar to the Internet application protocols such as HTTP [RFC2068] and SMTP [RFC821]. These interfaces are examples of what the simplest standard interfaces could look like.

A platform prototype has been implemented to demonstrate the IIN design, and a group of sample services have been implemented to demonstrate the possibilities of the proposed integration and the functioning of the platform prototype. The platform prototype is restricted to providing services which use communication over the Internet solely as a means of control in the IN, rather than for telephone-network-style content exchange (voice, fax etc.) with the telephone network. The implemented sample services are, also restricted to the "control-only-over-Internet" type, while the platform design does consider the potential provision of some other types of services.

1.2 Summary of Thesis

In **Chapter 2**, the Intelligent Network (IN) is defined and its most important aspects described in some detail. After an introduction in §2.1, §2.2 looks at what the Intelligent Network actually is, both from the aspect of the motivation and objectives behind its definition and development and from the aspect of the resulting group of characteristics and services of the IN. Section 2.3 gives a short account of the history of IN, while §2.4 describes IN standardisation activities, with an accent on the standards developed by the International Telecommunication Union (ITU). The remaining sections of Chapter 2 give more detailed descriptions of IN architecture and functionality, namely, descriptions of the Intelligent Network Conceptual Model (INCM), of IN functionality contained in the most basic IN – Capability Set 1, as well as brief descriptions of the other IN Capability Sets. A conclusion to the chapter is given in §2.8.

Chapter 3 describes computer networks and the existing types of computer network – telephone network integration. Section 3.1 defines and describes the different types of computer networks in existence, also giving a brief history of computer networks to date (§3.1.4). Section 3.2 elaborates on the Internet, its structure, functionality and mechanisms for developing Internet applications. Those include the sockets API, CORBA and Java. Section 3.3 investigates computer and telephone technology integration in the domain of private computer networks and private branch exchanges (PBXs), known as Computer-Telephony Integration (CTI). This section contains a general outline of CTI, the existing standards relating to CTI as well as an conceptual-level comparison between ECMA's (European Computer Manufacturers Association) Computer Supported Telephony Applications (CSTA) standard and the ITU IN standard. Section 3.4 contains information on existing

integrated systems composed of the Internet and a public telephone network (which may or may not be an IN) and applications for such systems. Section 3.5 concludes the chapter in a brief summary and an account of its relevance to the project.

Chapter 4 moves on to deal with the definition of the Internet-Integrated Intelligent Network (IIIN) that, together with the system prototype implementation, comprises the original work done for this project. After an introduction in §4.1, §4.2 introduces the Internet Integration Capability Set Increment 1 (IICSI1), the group of basic IN capabilities that have been identified as suitable for specifying an IN to support the most basic, "first generation" IIIN. Section 4.3 defines the IICSI1 in the context of the INCM, analysing its constituent capabilities in each of the INCM functional planes, using a set of IICSI1 Service-Independent Building Blocks (SIBs) as a formal means of describing those capabilities. Section 4.4 identifies the capabilities of the Internet necessary for providing IIIN services based on IICSI1. These capabilities are described in terms of Internet Application Service Components (IASCs), viewed from each of the four planes of the INCM. In the final section of Chapter 4, §4.5, various deployment considerations for the IIIN are listed and analysed briefly.

Chapter 5 is an account of the implementation undertaken as part of this project. Section 5.1 gives a general introduction, while §5.2 lists and describes the equipment and software tools used in the implementation. The description of the implementation itself has been laid out in two sections, following the strategy of implementation, with a basic IN prototype as an intermediate goal and an IIIN prototype, built to extend the IN prototype, as the ultimate goal. Therefore, §5.3 deals with the prototype of the basic IN, describing the implementation model, object models for each module of the implementation model and how the SIBs and services fit into those models. Further in that section (§5.3.2) two sample services are described in terms of their functionality and scenarios for their demonstration that have been successfully tested. Section 5.4 has a layout similar to §5.3, but deals with the IIIN prototype (which was built "on top of" the IN prototype) and with the IIIN sample services. Again the implementation model, object models for each module and descriptions of the SIB and service implementations are given. And once more the sample services are described in terms of their functionality and successfully tested demonstration scenarios. In addition to that, service specific functionality resident in the Internet has been described for each sample IIIN service. The final section, 5.5, describes the extent and results of the implementation.

Finally, **Chapter 6** summarises the work done and analyses the proposed system in terms of its applicability and usefulness in the "real world" as well as how the principles and facts stumbled upon during work on this project might be utilised in future research.

This thesis contains seven **appendices**. Appendix A contains the Stage 1 description of the newly introduced SIBs, while Appendix B contains their Stage 2 description. Appendix C comprises figures of SIB and IASC chains for all the IIIN services described in Chapter 4. In Appendix D, listings are given of the IDL and of the text-based protocol defined for application-level communication between the Internet-resident IIIN entities. Appendix E presents the implementation SIB-chains for the IN sample services. It also contains the implementation SIB and IASC chains for the cases of IIIN sample services for which those chains differ form the ones given in Appendix C. Appendix F elaborates the UML class diagrams for the prototype, shown in Chapter 5, in detailed UML descriptions of the classes. Finally Appendix G presents examples of C++, C and Java code of the prototype implementation

2 The Intelligent Network

2.1 Introduction

Supported and inspired by a wealth of advancements in information and telecommunication technology, services other that simple end-to-end connectivity have been developed and are now a seamlessly integrated part of modern telecommunication networks. The Intelligent Network (IN) employs technologies such as centralised and distributed data processing, digital switching and transmission techniques and complex signalling protocols to provide a standard platform for the development and deployment of such services.

The term "Intelligent Network", within the context of telecommunications today, refers to a specific architectural concept for telecommunication networks whereby diverse new services may be introduced into the network cost effectively and rapidly [Q1201]. It should be noted that telecommunication networks not complying with Intelligent Network architecture have provided "more than simple connectivity" services for a long time now and that, as far as functionality is concerned, the words "intelligent network", in their true sense, may rightfully be applied to such networks[1]. It was the growth of that added service functionality, and a subsequent need for optimising its introduction and utilisation, that prompted an effort towards defining the architectural concept, which now lays claim to the name "Intelligent Network".

Another factor that contributed to the Intelligent Network initiative was the deregulation – or the opening up to private businesses – of telecommunication service provisioning, which has been taking place in recent years in the industrialised world and was itself prompted by the proliferation of the telecommunication industry. It brings into the scene provisioning of diverse services by multiple parties, service customisation and service expansion, which all need to be supported by an appropriately structured network.

---

[1] It can also be applied to many other types of, non-telecommunications (in the narrowest sense), networks.

The crux of the IN concept lies in the idea of separating, logically and physically, the service control functionality from the network routing logic. The logical separation allows for the same service implementation to be used with various equipment from different vendors but at the same time, for players completely different from equipment vendors and network operators to provide their own services, using the existing network infrastructure. The physical separation brings about the sharing of service processing and information resources among many switching entities.

IN emerged and developed first in the USA and it later spread to the rest of the world, especially to Europe and Japan [Fayn96]. However, the development of the IN concept and standards at the international level has been (and still is) taking place within the Telecommunication Standardisation Sector of the International Telecommunication Union (ITU-T). This work concerns itself with IN primarily as defined in the ITU-T Recommendations, laid out in [Q1200].

2.2 A General View of the Intelligent Network

2.2.1 Intelligent Network Objectives

At a certain point in the history of telephone networks (namely, in the mid eighties) the need arose for the comprehensive definition of a new way of organising the network, which would include existing and new capabilities, to accommodate the changing technological, regulatory and market circumstances [Garr93]. Those circumstances were characterised by novel technologies, which incited and supported the emergence of new services, by a growing market need for various services, some of them tailored precisely to the needs of particular customers, and by the deregulation of telecommunications.

Network operators themselves initiated activities in the direction of defining a new architectural concept, which was to become the Intelligent Network. The main objectives of those activities were the following: first, to offer independence of network service provisioning from specific vendor equipment; next, to make possible fast and cost-effective introduction of new services into the network; finally, to create the opportunity for third parties (parties other than the network operator) to provide their own services over the network [Garr93].

2.2.2 Intelligent Network Characteristics

Here are a number of characteristics, which qualify the Intelligent Network as it has been defined to meet the above listed objectives [Q1201].

The implementation of services relies mostly on information processing techniques and is, therefore, made a purely software problem, which makes for faster and easier solutions. Basic network functionality is modularised and can be reused at service creation, which saves time as well as resources and also enables non-engineers to design and create services.

The communication between network functions is through standard, service independent interfaces, which gives freedom in service implementation coupled with the possibility of building a network using functional entities from many different vendors. The crucial interface of the IN is the one between the switching functions and the service control functions, which embodies the service processing independence from multiple-vendor equipment.

The service user and service subscriber (see §2.2.5) are given control over some service parameters specific to the user and subscriber, respectively.

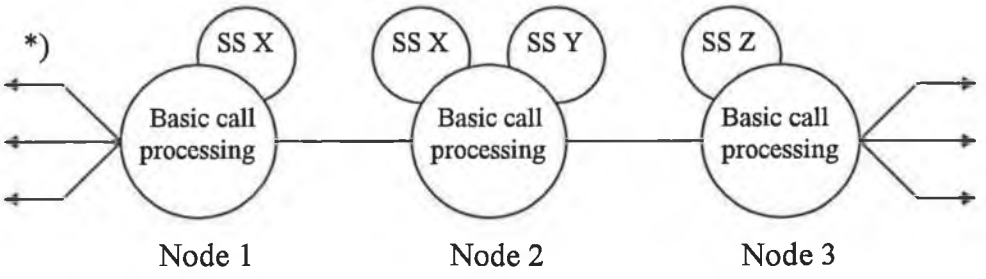
Network functions are flexibly allocated to and portable among physical entities. Network resources are used efficiently. The management of service logic is standardised.

2.2.3 IN Service Processing

This section seeks to demonstrate the most basic principle of the Intelligent Network - the service independence from the underlying network - by briefly describing the way service processing takes place in IN.

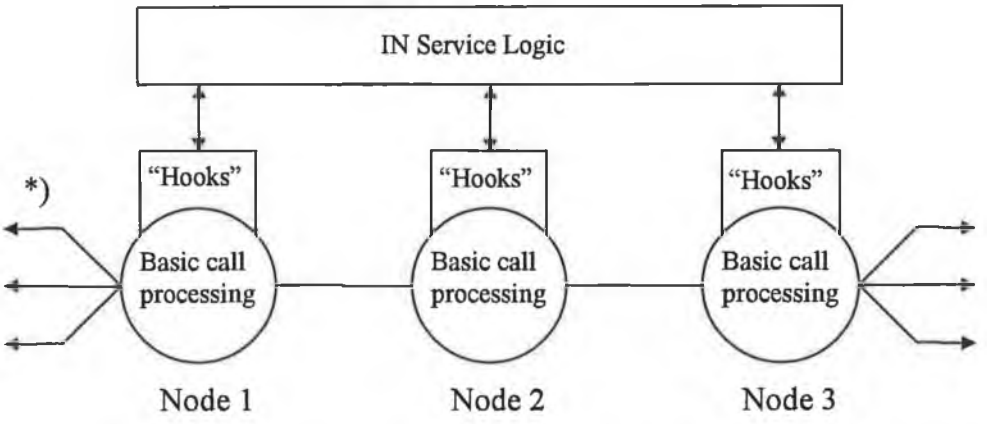
Figures 2.1 and 2.2, which can be found in [Q1201], depict models for non-IN service processing and IN service processing, respectively. In the non-IN service processing model, supplementary service processing functionality is implemented as and add-on to the basic call processing functionality and its implementation and deployment have to be tightly coupled with that of the basic call processing functions. In the IN service processing model, however, the basic call processing function is equipped with "hooks". These hooks are in fact points in the basic call processing procedures at which those procedures are suspended and a dialogue is started with the IN service logic, provided certain conditions, which indicate the need for a supplementary service, are met. Once the IN service logic has finished processing the service, the basic call processing resumes. If such a point is encountered during the

basic call process and the relevant conditions are not met, the processing continues undisturbed. The points in the basic call, at which the processing can be suspended, are carefully chosen and standardised, as is the communication between the basic call process and IN service logic. Thus various services can be implemented independently from the network providing basic call functionality, by anybody familiar with the communication interface – and deployed, by anyone with access to a "live" interface.



SS - Supplementary Service
*) - Basic and supplementary services interface towards customers

Figure 2.1 - Non-IN Service Processing Model



*) - Basic and supplementary services interface towards customers

Figure 2.2 - IN Service Processing Model

2.2.4 Scope of the Intelligent Network

Although at first developed to meet the needs for new service introduction in the public switched telephone network, the IN concept is seen as applicable to several different types of networks. They include systems, such as mobile communication systems, which inherently require intensive call processing capabilities and for which IN seems a natural solution. Types of telecommunication networks regarded to fall

within the scope of IN are: public switched telephone network (PSTN), mobile network, packet switched public data network (PSPDN) and integrated services digital network (ISDN) – both narrow-band (N-ISDN) and broad-band (B-ISDN) [Q1201].

2.2.5 Roles in the Intelligent Network

The many actors who take part in some way in the activities within the Intelligent Network can be divided into a number of categories, as identified in [Ovum93]. First, there is the *network operator*, who owns the infrastructure and offers services in the telecommunication network. The *service provider* provides additional services over the telecommunication network. The *service subscriber* is an organisation or person who buys a service, which could be customised to his particular needs, and offers it to the users in the network. The *service user* uses the service offered by the service subscriber. For example, a bank gets a 1800 (freephone) number from the service provider and so becomes a service subscriber. Anybody who rings that number to contact the bank is a service user in that particular instance.

2.2.6 Intelligent Network Services

A common example of and IN service is the Freephone service. Apart from allowing a subscriber to be billed for incoming calls, it can also include specially designed routing, whereby calls are routed to different subscriber premises, depending, for example, on the time of day or day in the week. The IN has to perform number translation (from an 800 or 1800 number to a real address), make special billing arrangements (the called party is billed instead of the calling) and make a decision as to where to route the call.

Another example is the Virtual Private Network (VPN) service, whereby a business subscriber is allowed to view all the communications within the company, even if they are between two different geographic locations and are maintained through the public network, as communications on a private network. The IN achieves this by translating specially assigned VPN numbers to public network numbers, when the call is originated from a telephone belonging to the VPN to a telephone in the same VPN.

Other services are Account Card Calling (the user uses the card and a PIN to access the network and is billed to a special account for any calls made in that way),

Televoting (the users can dial a number and place a vote after listening to the choices, by sending a DTMF digit – used by television and radio stations) and Universal Personal Telecommunications (the user uses a Personal Telecommunication Number (PTN) to access and receive calls at any location and in any type of network).

IN is also suitable for providing mobile communication services and broadband services. Even simple connectivity in a mobile network can make use of IN since it involves a lot of processing in order to locate terminals, follow them during calls etc. It is also suitable for broadband networks, where a call may involve several different media and several parties. Apart from that, the existing IN services can be provided over either of these two networks.

2.3 History

The first services based on the Intelligent Network (IN) architectural concept appeared in 1981 and it was not until 1986 that the actual phrase "Intelligent Network" was associated with the concept, in Bellcore. However, the roots of IN lie in the 1960s, when the first Stored Program Control (SPC) exchanges were deployed, namely, AT&T's ESS switches, introducing computer technology into the telecommunication network. These systems could support services such as Call Waiting, Call Forwarding and Centrex on a local level [Fayn96].

A service called Inward Wide Area Telecommunications Service (INWATS), which was a rudimentary "800 number" service, was available in the long-distance network already in the late sixties, on crossbar switches, but the implementation, as one would imagine, was complicated and cumbersome. The first electronic switch in the AT&T long distance network was introduced in 1976. The crucial innovation for IN, however, was common channel signalling, which was first deployed in the mid-seventies, both in the US (under the name Common Channel Interoffice Signalling or CCIS) and in Europe (called Common Channel Signalling Number 6 or CC#6). At first this system was used only for the transmission of address digits and trunk status information (instead of tone signals) but it was soon proposed that the signalling system be used for communication between a network database and the switches in the network.

This idea came to fruition when two services based on it, the Calling Card and an "800 number" service (CCIS INWATS) were deployed in 1980 and 1981, respectively. At this point work began in Bell Labs towards creating a general platform with primitive messages between the switch and the service processing

entity, which could be used to create different services. The result of these activities was the Direct Service Dialling Capabilities (DSDC) architecture and the first services implemented on it were the 800 service and Software Defined Network (SDN), the predecessor of the Virtual Private Network (VPN) service.

In Europe, France and Germany introduced the freephone service in 1983 and the UK in 1985, using AT&T products. In Japan, NTT implemented its own freephone service in 1985.

In 1984 Bell System was dissolved into several Regional Bell Operating Companies (RBOCs) and Bellcore (the research company associated with the RBOCs) began work on the total separation of service features from the switching process, particularly influenced by the RBOCs' multiple-vendor equipment supply. This first IN endeavour on the local network level resulted in a version of IN called IN2 which, being very ambitious, was never realised. Then the Multivendor Interactions Forum was formed to include the many equipment vendors in the development process. What came out of it was the Advanced Intelligent Network (AIN) which is now being implemented in the US in small steps (AIN 0.1 was launched in 1992) [Fayn96].

In 1989 the International Consultative Committee for Telegraph and Telephone (CCITT, now ITU-T) started work on the standardisation of IN. Today many European countries have implemented and deployed IN based on the result of the first phase of specification, the Capability Set 1 (CS-1), the ITU-T recommendations for which are listed in [Q1210].

2.4 Intelligent Network Standardisation

2.4.1 Introduction

There are separate standardisation activities in the area of Intelligent Networks (IN) in many countries. Apart from creating national standards, they are also aimed at contributing to the standardisation process in the Telecommunication Standardisation Sector of the International Telecommunication Union (ITU-T), which was started in 1989. The European Telecommunication Standards Institute (ETSI) works both on the development of European IN standards and on preparing propositions for the ITU-T.

While in the United States the development of IN took place in the direction from the field and innovation of real systems towards standardisation, in Europe, where IN was not introduced until later, it took a somewhat opposite direction, with many realised systems being based on the ITU-T standards. For that reason, the ITU-T standards, listed in [Q1200], are particularly interesting for us.

2.4.2 ITU-T Recommendations

A clarification: the documents referred to above as ITU-T Intelligent Network standards are formally only ITU-T Recommendations and it is left to each country to turn such recommendations into actual standards. This, however, does not affect the way the contents of the documents is viewed and interpreted.

The plan of the ITU-T is to specify IN in phases. Each phase produces a Capability Set which is an increment of the previous phase and in its implementation is backward compatible. The first Capability Set (CS-1) encompasses a well defined set of services and a specific part of IN functionality. The set of services and part of the IN functionality that are specified and implemented are increased with every Capability Set.

The IN is standardised in Recommendations Q.12xy, where x and y are both any digit. Recommendation Q.120y (x = 0) are general recommendations on different aspects of IN. Where x is between 1 and 8, the Recommendation is on some aspect of a specific Capability Set designated by x. For example, Q.1211 and Q.1214 pertain to CS-1. The digit y designates the specific aspect of IN, however, those are elaborated on in the further sections of this chapter. Recommendation Q.1290 is a glossary of IN terms.

The CS-1 Recommendations were formally approved in 1993 and there are widespread implementations of IN based on them. The work on CS-2 was published in 1998 and is generally not yet implemented. Work on CS-3 is in progress.

The rest of this chapter focuses on the Intelligent Network specification laid out in the general and CS-1 Recommendations of the Q.12xy series, while also including references to the CS-2 Recommendations.

2.5 The Intelligent Network Conceptual Model

2.5.1 General

The Intelligent Network Conceptual Model (INCM) has been, as its name implies, designed to incorporate the *concepts* which define the Intelligent Network (IN) and to thus serve as a model for describing Intelligent Networks and developing implementation specifications for them. In the INCM, the IN concepts are put into the context of the overall Intelligent Network. Thus the concepts are concretised and their relations to other concepts clarified.

The INCM offers four different views of the IN, each on a different level of abstraction – called a "plane" – and defines the mappings between those views. The planes could be seen as perspectives of the Intelligent Network pertaining to different actors in the IN and the service deployment process. Figure 2.3 shows the INCM [Q1201].

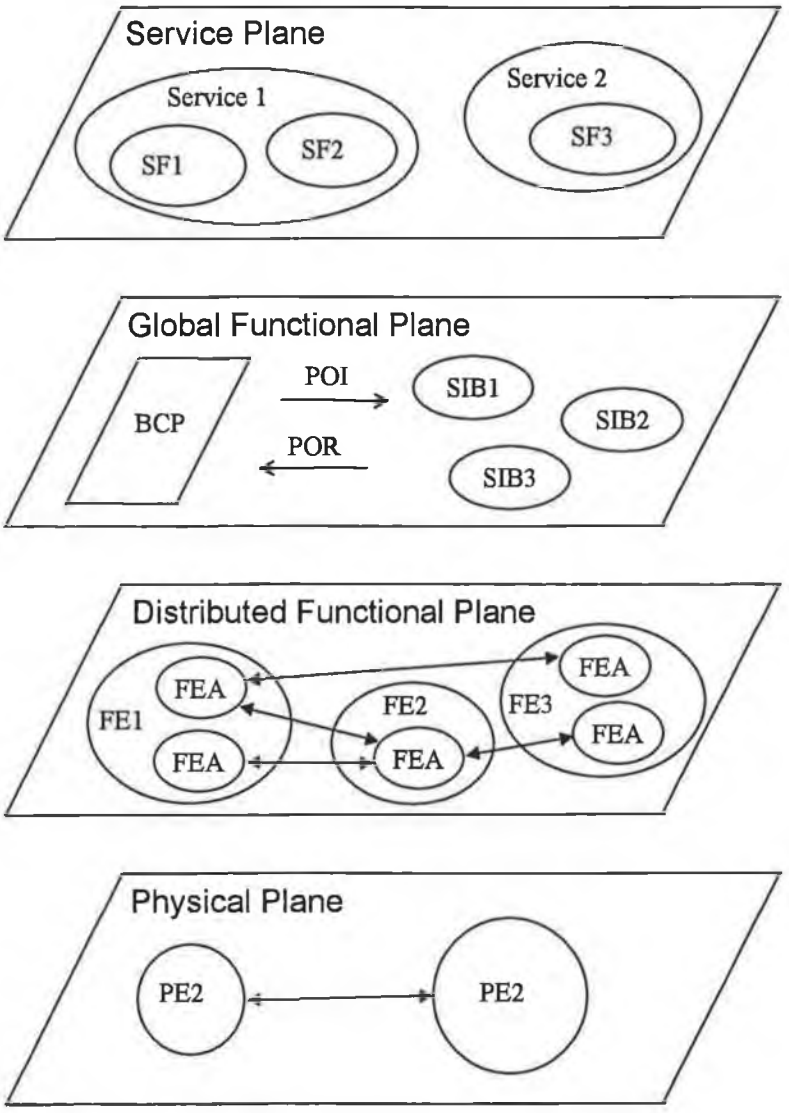
2.5.2 Service Plane (SP)

This is the uppermost plane of the INCM, which offers the most abstract view of the Intelligent Network. All that can be seen at this level are services, as would be described to or by the user. There is no indication of how the services are implemented. The services are composed of one or more service features, which may be either core features that carry the main functionality of the service, or optional parts offered as enhancements to the telecommunication service [Q1202].

2.5.3 Global Functional Plane (GFP)

This plane presents a view demonstrating the IN feature of service creation by putting together modules of reusable network functionality. There is no view of the network and its details. What can be seen are modules of functionality, provided by the network as a whole and the way these modules are put together to create services. The "modules of functionality" are called Service Independent Building Blocks (SIBs). The Basic Call Process (BCP), which is invoked for all calls, IN or not, and handles basic connectivity, is considered to be a special SIB [Q1203].

The SIBs themselves are described formally, with service specific and instance specific parameters and with the possible outcomes and results of their execution, as well as errors that may occur.



SF - Service Feature
BCP -Basic Call Process
POI - Point of Initiation
POR - Point of Return
SIB - Service-Independent Building Block
FE - Functional Entity
FEA - Functional Entity Action
IF - Information Flow
PE - Physical Entity
P - Protocol

Figure 2.3 - IN Conceptual Model

2.5.3.1 Mapping of Service Plane

A service is described in terms of where the BCP should be interrupted (see §2.2.3) to invoke the processing of that service (the Point of Initiation or POI), the chain of SIBs that need to be executed, and the point at which the BCP is resumed once the IN-specific processing is finished (the Point of Return or POR). Figure 2.4 illustrates service description on the Global Functional Plane [Q1203].

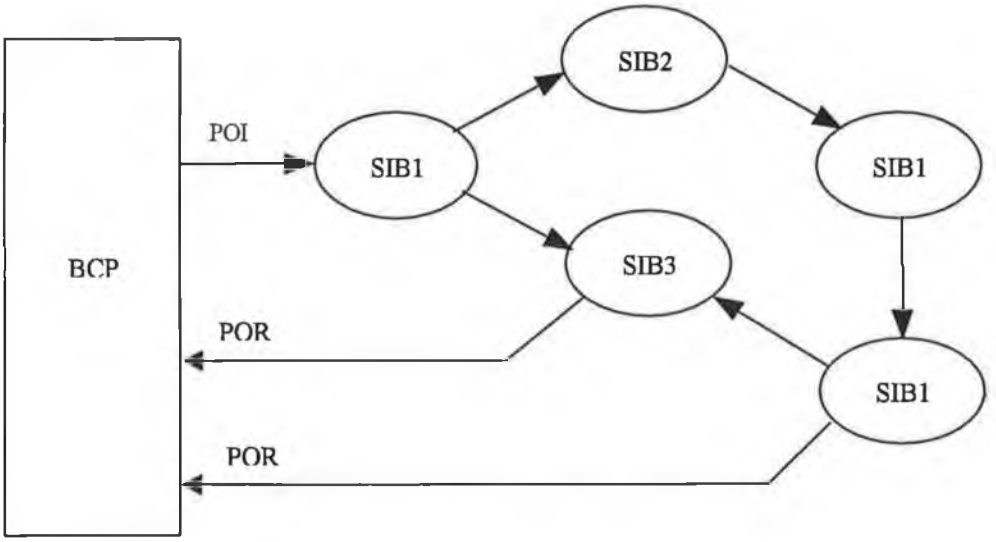


Figure 2.4 – SIB Chain Representation of a Service

2.5.4 Distributed Functional Plane (DFP)

This plane offers a view of the network as a collection of Functional Entities (FEs). The physical location of these functional entities is not visible. What is seen here is the distribution of the functionality in the network. Each FE can perform a number of Functional Entity Actions (FEAs) and the FEs communicate by sending Information Flows (IFs) to each other. The FEAs are further broken up into Elementary Functions (EFs) [Q1204].

2.5.4.1 Mapping of the Global Functional Plane

The SIB functionality is spread over several FEs, performing Functional Entity Actions and communicating through Information Flows. A SIB is mapped onto a sequence of IFs and FEAs and can be represented in the form of several message sequence charts or an SDL diagram [Q1204].

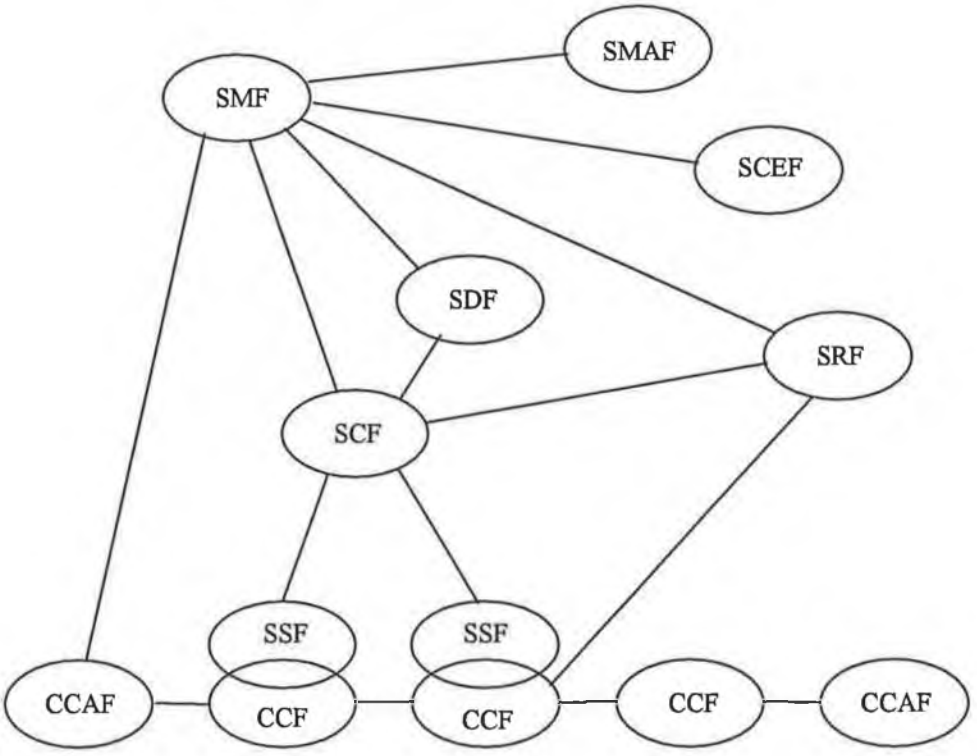
2.5.4.2 IN Distributed Functional Plane Model

The concrete Functional Entities of IN and the relationships between them are shown in Figure 2.5, which depicts the IN DFP Model [Q1204].

The **Call Control Agent Function (CCAF)** provides the user with access to the network and serves as an agent between the user and the service-providing capabilities of the Call Control Function.

The **Call Control Function (CCF)** carries the basic call and/or connection control functionality, acting at the requests of the CCAF. It provides triggers for accessing IN functionality.

The **Service Switching Function (SSF)** processes the triggers received from the CCF to recognise service control triggers. It manages the signalling between the CCF and the SCF and modifies, when necessary, the call processing in the CCF as requested by the SCF. The SSF and the CCF are tightly coupled and are always implemented together. The reason they are separate entities is that the SSF is used to represent the IN-specific part of the switching functionality. CCF, on the other hand, represents the non-IN, basic call processing functionality.



CCAF - Call Control Agent Function
CCF - Call Control Function
SCEF - Service Creation Environment Function
SCF - Service Control Function
SDF - Service Data Function
SMAF - Service Management Access Function
SMF - Service Management Function
SRF - Specialised Resource Function
SSF - Service Switching Function

Figure 2.5 - The IN Distributed Functional Plane Model

The **Service Control Function (SCF)** is the central Functional Entity of the IN. It controls the execution of IN-implemented services. It performs the service processing but also interacts with the SDF to get or set network-stored information required by the service and with the SRF when special resources are required. Indeed,

it communicates with the SSF from which requests for processing arrive and to which the SCF sends related call control instructions.

The **Service Data Function (SDF)** contains customer and network data and is queried by the SCF during service processing.

The **Special Resource Function (SRF)** contains specialised hardware resources required for IN services, mainly for user interaction, such as announcement players, tone generators, voice recognition and digit collection devices, but also others such as text to speech synthesisers, protocol converters and conference bridges. It is controlled during service processing through interactions with the SCF.

Following is the description of *IN Functional Entities related to service creation/management*:

The **Service Creation Environment Function (SCEF)** provides an environment for the definition, development, testing and online deployment of services.

The **Service Management Access Function (SMAF)** is the equivalent of the CCAF, but in the management domain of the IN. It provides managers with access to the SMF.

The **Service Management Function (SMF)** allows the deployment and provision of IN services and, generally, manages and updates most of the other FEs (CCF, SSF, SCF and SDF). It also provides "online" management such as the collection of billing and statistic information.

2.5.5 Physical Plane (PP)

On the physical plane a full view of the physical network is available. It shows the Physical Entities (PEs) present in the network and where the functional entities are located, in terms of physical entities. An important aspect of the view of IN on the physical plane is the definition of protocols, which are used for communication between the physical entities, depending on which functional entities they contain.

2.5.5.1 The Physical Entities

The **Service Switching Point (SSP)** is, in fact, a switch and contains the CCF and the SSF. In case that it is a local exchange, it also contains the CCAF.

The **Service Control Point (SCP)** contains the SCF but may also contain the SDF.

The **Service Data Point (SDP)** contains the SDF.

The **Intelligent Peripheral (IP)** contains the SRF and possibly a CCF/SSF, to provide external access to its switching matrix.

The **Service Management Point (SMP)** contains the SMF and possibly the SMAF, in the case where the manager is directly working on the machine that is the SMP. It may also include the SCEF.

The **Service Creation Environment Point (SCEP)** contains the SCEF.

The **Service Management Access Point (SMAP)** contains the SMAF. It is a point of access for a manager to the SMP, which contains the SMF. It acts like a terminal attached to the SMP.

There are three other physical entities defined for the Intelligent Network, which do not have a one-to-one mapping to a functional entity, as do those listed above.

An **Adjunct (AD)** more or less like a SCP except that it is directly connected to a SSP and not through the signalling network, which may result in much faster communication between the two entities.

A **Service Node (SN)** is similar to the adjunct in that it is directly connected to the SSP. However, apart from the SCF, it contains a SDF, a SRF and a CCF/SSF. The CCF/SSF is not accessible from the outside of the SN. The SRF is accessible from the outside of the SN and the SN can be used as any other IP by any SCF.

A Service Switching and Control Point (SSCP) combines the SCF and SSF in one node. It contains the SCF, SDF, CCAF, CCF, SSF and, possibly, the SRF. Although the functionality provided is the same as that of a separate SSF and SCF, the interfaces between those functional entities within the SSCP are proprietary [Q1205].

2.5.5.2 Communication between Physical Entities

The protocol used for communication between physical entities in IN is the Intelligent Network Application Protocol (INAP). It is a user of the Remote Operations Service Element (ROSE) protocol, which contains primitives suitable for performing operations between physically remote entities and is standardised through ITU-T recommendations X.219 and X.229. ROSE, in its turn, is contained in the upper layer of the Transaction Capabilities Application Part (TCAP) [Q771], which,

for the needs of IN, may use the lower layers of the SS7 signalling stack or some other protocol, for network layer functionality [Q1208].

2.5.6 Summary

As can be seen from the previous sections, the INCM, in spite of its four-plane anatomy, does not create further confusion as to what IN actually is. It puts the defining features of IN into context and clarifies their meaning within IN, while also showing the relations between the separate features. The four planes simplify the view of a complex, multidimensional concept.

The service plane offers a general view on the IN service capabilities that is, what IN has to offer to the customer. The global functional plane demonstrates the IN principle of modularity and reusability of network functionality in service creation. On the distributed functional plane, the network is presented in terms of functionally distinct entities. Especially important is the separation between the switching and the service control function, which embodies the principle of service provisioning independence from the underlying network. The existence of the SMF and the SCEF makes the IN a service creation and deployment platform, which, besides being an incarnation of the INCM, must have a view of itself in terms of the INCM in order to allow for the construction of services. The physical plane demonstrates the flexibility of function distribution among actual physical entities and the portability of functions among those entities. The requirement for standardised interfaces between physical entities allows for the use of equipment from multiple vendors in IN.

2.6 Capability Set 1

2.6.1 Introduction

Capability Set 1 (CS-1) is the result of the first phase of IN specification by the ITU-T. The ITU-T Recommendations pertaining to that capability set were first published in 1993. They contain the description, from the point of view of each functional plane, of the capabilities which should be offered by IN in the first phase of its development.

CS-1 is restricted to services, which contain only *single-ended, single point of control* service features[2]. A single-ended service feature is one that deals with only one party in a call. The other parties in the call may be associated with other instances of the same feature or other features provided those do not interfere with the feature pertaining to the first party. Single point of control means that no more than one SCF can be controlling the same aspect of a call at the one time. These restrictions were introduced mainly to avoid dealing with feature interaction, which is a complex problem in itself and would require a comprehensive solution.

CS-1 pertains solely to the service execution aspect of IN, describing the possible services as well as the SIBs, functional entities, interfaces and procedures within the functional entities needed for the implementation and execution of the delimited set of services. It does not address IN service management or service creation capabilities.

Among existing telecommunication network types PSTN and narrow-band ISDN are the only ones covered by the CS-1 specifications.

This section lays out CS-1 capabilities appropriately on each of the planes of the INCM, i.e. in the form of service descriptions, SIBs, distributed functionality and interfaces between functional entities. At the same time, it demonstrates the methodology used for these descriptions and specifications, which will be used for presenting the IIIN design further in this thesis.

2.6.2 Service Plane

On the service plane, CS-1 consists of a target set of 25 services and 38 service features. Each service includes one or more service features. Some of those service features form the core of the service, while others are optional and may be added to the service in order to enhance it.

Two examples of CS-1 services are Abbreviated dialling and Call forwarding [Q1211].

The Abbreviated dialling (AD) service has one core service feature, also called Abbreviated dialling (AD), and two optional service features, Call logging (LOG) and Customer profile management (CPM). AD is a service feature whereby users can dial a short digit sequence instead of the whole number when calling another party. Call logging allows for a record to be prepared each time a call arrives at a specific telephone number. Customer profile management allows the subscriber

[2] This also applies to CS-2 service features.

to manage his/her service profile in real-time, and can be included in the Abbreviated dialling service to allow the customer to specify some numbers and sequences to replace them.

The Call forwarding (CF) service has one core service feature, also called Call forwarding (CF), and the same two optional service features as Abbreviated dialling, LOG and CPM. The Call forwarding service feature allows the customer to have all his/her calls addressed to another number. If the CPM is included as a service feature in the CF service, the customer herself can set the number to which she wants to have the calls forwarded.

2.6.3 Global Functional Plane

The GFP describes SIBs as "black boxes", with a logical start, logical end, input and output parameters. Fourteen SIBs have been described in CS-1, as well as the Basic Call Process (BCP) as a special case SIB [Q1213].

2.6.3.1 Basic Call Process (BCP)

The BCP can be called a SIB in that it is a functional building block, in fact, a building block present in every service. However, it is fundamentally different from the other SIBs in two ways. First, the BCP functionality is non-IN call-control functionality, as viewed by the IN, while all the other SIBs perform specifically IN tasks. Second, it is activities in the BCP (which exists for every call in the IN, whether associated with an IN service or not) that invoke service logic – and therefore SIBs – when an IN service is requested.

From the IN point of view, the BCP is most markedly characterised by Points of Initiation (POI) and Points of Return (POR). POIs are specific points in the BCP at which the processing is suspended and a request issued to the service processing logic, provided certain conditions are met. PORs are returning instructions from the service processing logic to the BCP with instructions on how to continue processing.

The POIs are Call Originated, Address Collected, Address Analysed, Prepared to Complete Call, Busy, No Answer, Call Acceptance and End of Call. The PORs are Continue with Existing Data, Proceed with New Data, Handle as Transit, Clear Call, Enable Call Party Handling and Initiate Call.

The Initiate Call POR is not really a point of return to the basic call process. It is, in fact, an instruction from the service logic to the switching entity to start a new basic call process. Such a capability somewhat disturbs the IN service processing

model but has been included in CS-1 as a valid means of starting a BCP-IN service logic interaction[3].

2.6.3.2 SIBs

Fourteen CS-1 SIBs have been defined [Q1213]. They are:

ALGORITHM,	LIMIT,	STATUS NOTIFICATION,
AUTHENTICATE,	LOG CALL INFORMATION,	TRANSLATE,
CHARGE,	QUEUE,	USER INTERACTION
COMPARE,	SCREEN,	and
DISTIRBUTION,	SERVICE DATA MANAGEMENT,	VERIFY.

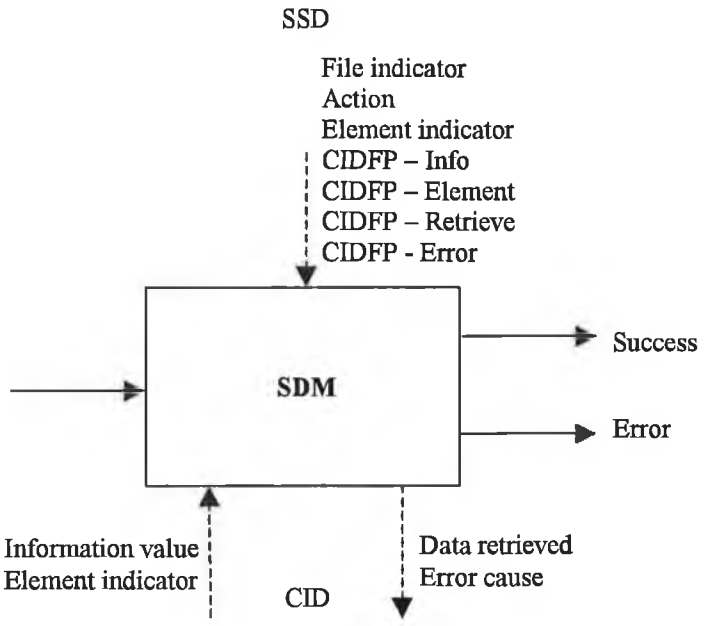
At the global functional plane the SIBs are described as modules of functionality and no details are given on what part of the network performs that functionality and in what way. What is given, is a description of the SIB's functionality and its applicability to services. The possible logical ends to the SIB execution and input and output parameters are also specified. The input data are of two kinds: Service Specific Data (SSD), which are the same for any instance of a particular service containing the SIB, and Call Instance Data (CID), which have instance-dependent values. All the output data are, naturally, of the CID kind. Here are two examples of GFP SIB description, which is also referred to as *stage 1 SIB description*.

The SERVICE DATA MANAGEMENT (SDM) SIB has the capability of manipulating user-specific data. It updates and retrieves user data from the database. The retrieval of user-specific data is a part of many different services, such as Call Forwarding (the database stores the number to which the user's calls are to be forwarded) and Credit Card Calling (the user's personal details are kept in the database). The update of the database can take place during service execution or during management activities (this is the only SIB which performs service management functions). The SIB is graphically represented in Figure 2.6.

The SSD for this SIB consist of a pointer to the object in the database. The input CID consist of: the action to be performed (read, write etc.), the element of the object that should be accessed, a pointer to a value used for incrementing and decrementing (if that is the action required) and a pointer to the value of the element

[3] This capability fits into CS-2 much more organically, as it is supported by the CS-2 Call Party Handling capabilities [Q1224].

(for the case of writing). The output CID consist of a pointer to the retrieved value and a pointer to the error value (for the case when an error occurs in the SIB execution).



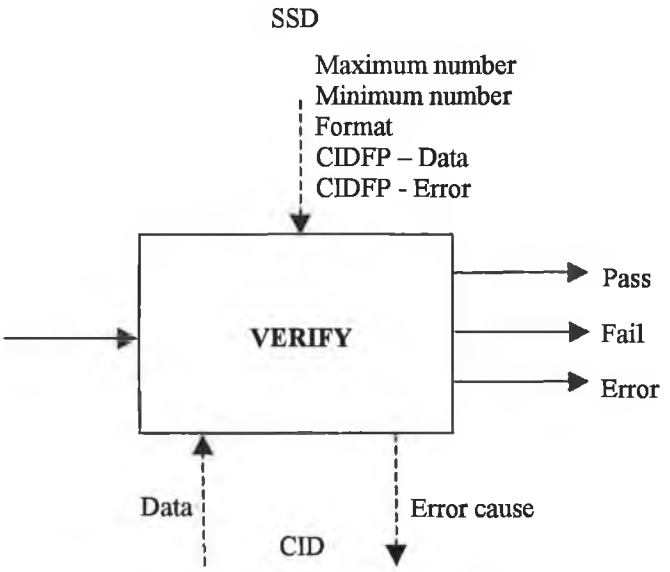
SDM – Service Data Management
SSD – Service Support Data
CID – Call Instance Data
CIDFP – Call Instance Data Field Pointer

Figure 2.6 – GFP Description of the SDM SIB

The Logical Ends for this SIB are Success and Error (in which case, the error output CID value is set).

The VERIFY SIB is usually used after the USER INTERACTION SIB to verify the syntactical correctness of data collected from the user. It is applicable to any service which engages in digit collection from the user. For example, when invoking the Account Card Calling service the user must enter a Personal Identification Number (PIN), which then needs to be verified. The graphic representation of this SIB is given in Figure 2.7.

The VERIFY SIB's SSD consist of the values for the maximum and minimum allowed length of the string to be verified, and of the specification of a format which should be matched by the syntax of the input data. The input CID consist of one parameter, a pointer to the data string which is to be verified. The output CID data comprise a pointer to an error value (for the case that an error occurred during the execution of the SIB).



SSD – Service Support Data
CID – Call Instance Data
CIDFP – Call Instance Data Field Pointer

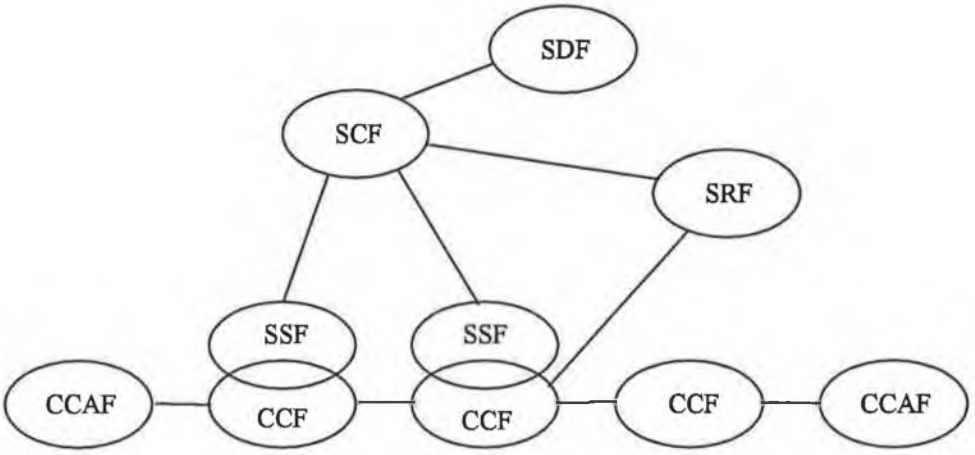
Figure 2.7 – GFP Description of the VERIFY SIB

The Logical Ends for this SIB are Pass, Fail and Error. In the case of the Error Logical End, the error parameter in the output CID is set to a value depending on the type of error.

2.6.4 Distributed Functional Plane

On this plane, details of the distribution of call processing functionality in the network are given, as well as the inside architectures of the functional entities relevant to CS-1. A set of functional entity relations and internetworking aspects tackled by CS-1 is also specified. The part of the Distributed Functional Plane Model dealt with in CS-1 is shown in Figure 2.8 [Q1214]. The definitions of the Functional Entities are the same as in §2.5.4.

This section discusses the most important aspects of the SSF, briefly describes the SDF and the SRF, while giving a detailed description of the SCF architecture. It shows the *stage 2 SIB description* (DFP SIB description) method and gives an example of it, at the same time introducing Functional Entity Actions (FEAs) and Information Flows (IFs).



CCAF - Call Control Agent Function
CCF - Call Control Function
SCF - Service Control Function
SDF - Service Data Function
SRF - Specialised Resource Function
SSF - Service Switching Function

Figure 2.8 - The IN Distributed Functional Plane Model for CS-1

2.6.4.1 Call Control Function / Service Switching Function

The scope of this work does not require a detailed examination of the CCF/SSF architecture. The CCF part consists of a Basic Call Manager (BCM) which has access to bearer control and to the CCF resource manager. At the same time, the BCM is linked to the SSF, which has multiple functions such as providing the SCF with a high-level view of the call/connection processing activities, processing events coming from the BCM and reporting them to the SCF if the right conditions are met, managing SSF resources and detecting and handling possible service feature interactions [Q1214].

For us, the most interesting aspect of the CCF/SSF is the functionality of the BCM. The BCM handles the basic call processing in IN. The Basic Call Process (which is the process controlling setting up, maintaining and breaking down of an end-to-end connection) is described by the Basic Call State Model (BCSM).

2.6.4.1.1 The Basic Call State Model (BCSM)

The BCSM, in fact, consists of two parts, one for the originating part of the call and the other for the terminating part of the call, O_BCSM and T_BCSM respectively. Figure 2.9 depicts the O_BCSM and Figure 2.10 depicts the T_BCSM [Q1214].

The BCSM (O_BCSM and T_BCSM) is built of two kinds of elements: Points In Call (PICs) and Detection Points (DPs). Points in call are activities of the Basic Call Process (BCP). For example, the Collect_Information PIC denotes the action of collecting digits from the user as he/she dials a number. Detection Points are points in the BCP at which it may, through the SSF, interact with the SCF. The O_BCSM has 6 PICs and 10 DPs, while the T_BCSM has 5 PICs and 7 DPs. They are all shown in Figures 2.8 and 2.9.

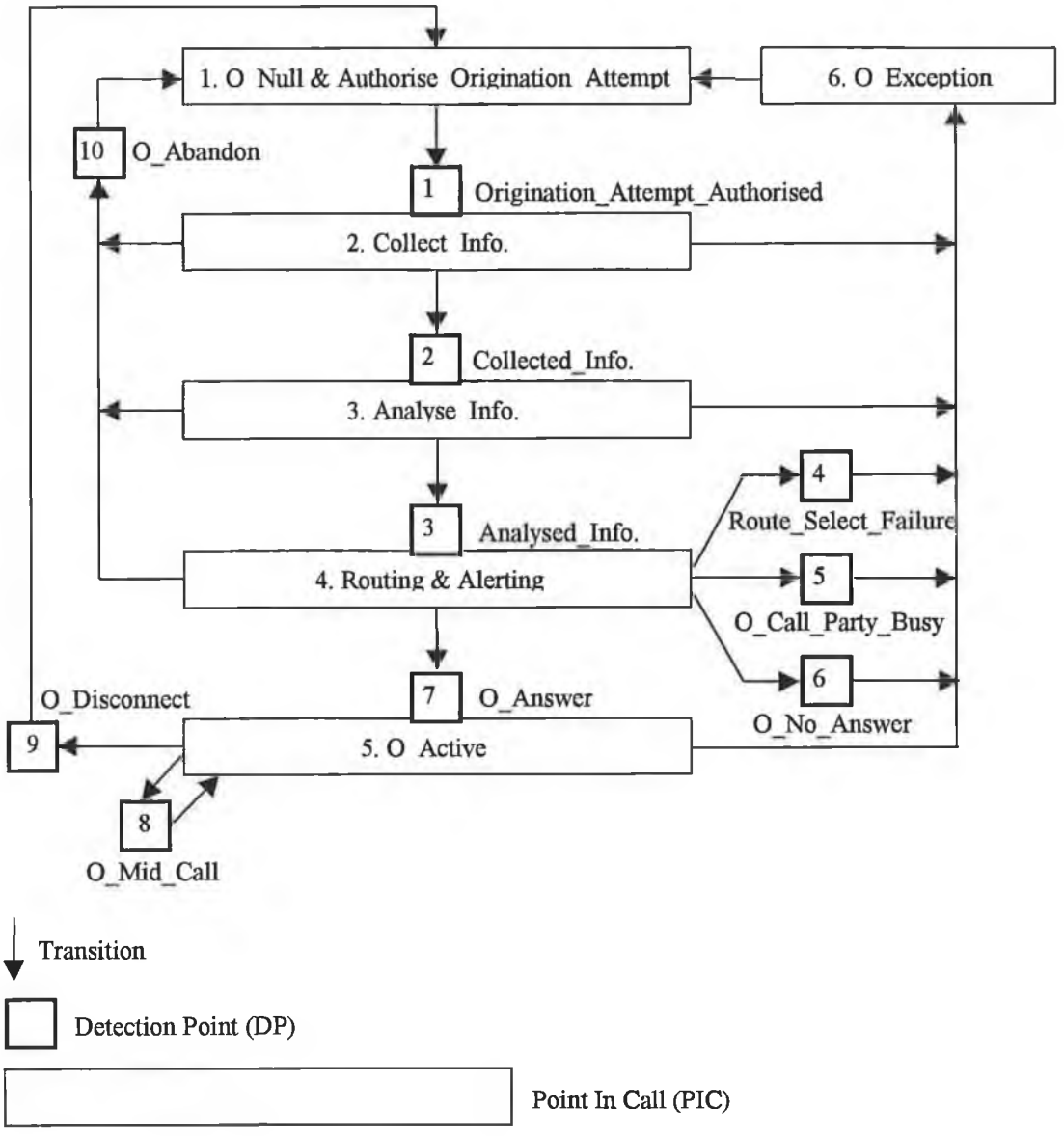


Figure 2.9 – Originating BCSM for CS-1

Whether the interaction takes place or not is decided according to certain criteria, which are also specified and vary from DP to DP. The one criterion, which must be met at any DP in order for a dialog with the SCP to start, is that the DP be

armed. Arming a DP basically means marking it as active. If the DP is not armed, it is ignored and the next PIC is executed. If the DP is armed, other criteria pertaining to that DP are examined to determine if a dialog with the SCF should be started or not. An example of a criterion is the *Specific Digit Strings* criterion, which, among others, applies to DP2 (Collected_Information). If DP2 is armed, the dialled digit string is examined and if it starts, for example, with 1800[4], the request is sent to the SCF for the translation of the number to a real address string.

The BCP resumes at a PIC, either the one following the DP at which it was suspended or at some other PIC, with the data it already has or new data supplied by the SCF, depending on the instructions by the SCF at the end of the dialog.

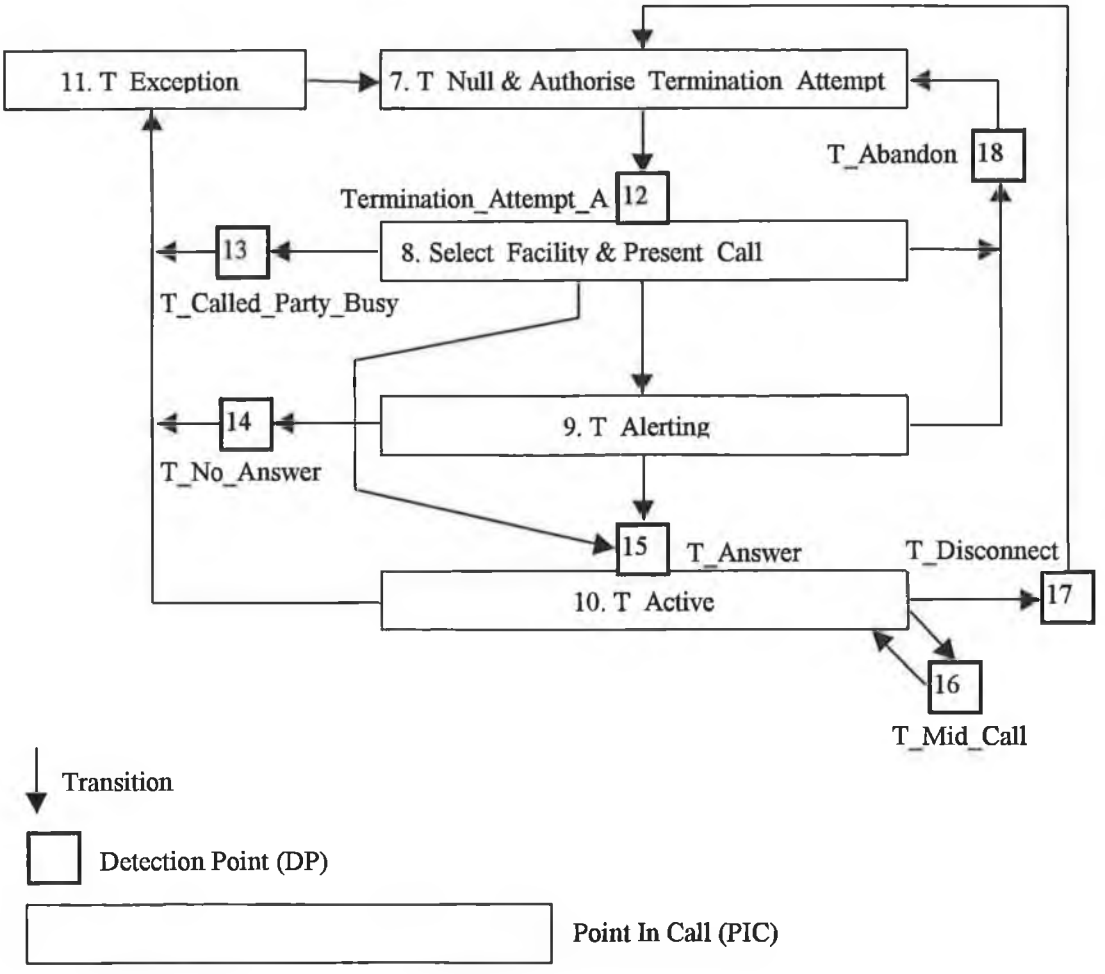


Figure 2.10 – Terminating BCSM for CS-1

It should be noted that the POIs of the Basic Call Process (see §2.6.3.1) can be mapped onto DPs, and PORs onto the different manners of BCP resumption. For example, the Address Collected POI corresponds to DP2 (Collected_Information) and

[4] The string denoting a freephone number in Ireland.

the Proceed With New Data POR corresponds to the continuation of processing at any[5] PIC, using data given by the SCF.

In some cases POIs have corresponding DPs in the O_BCSM as well as the T_BCSM. For example, the Call Acceptance POI maps onto DP7 (O_Answer) and DP15 (T_Answer). This happens because the BCP (of the Global Functional Plane) is monolithic and does not reflect the possibility of services being invoked from both sides of a call. Such a BCP is sufficient for CS-1 services, since they are single-ended (they deal with only one party of a call) and each can be described using a set of party-unspecific POIs, which will either pertain to the calling party or the called party, depending on the service. On the Distributed Functional Plane, the separation of the Originating and Terminating BCSMs displays the fact that services can be invoked from either side in the call. At the same time, however, it reflects the restrictions imposed by the Single-Ended principle of CS-1, whereby the manipulations of the call by the SCF-based service logic are consigned to the PICs and DPs of the part of the BCSM (Originating or Terminating) from which its activities were invoked.

2.6.4.1.2 Detection Point Processing

The described use of DPs is actually only one of the two existing ways of using DPs. As described above, a DP can be armed and associated with a certain number of criteria to provide a point for the beginning of interaction between the CCF/SSF and the SCF. DPs are thus either armed or not armed permanently, or until services associated with the DPs are introduced or withdrawn. This type of arming is called *static* arming and a DP so armed is called a Trigger Detection Point (TDP).

A DP can also be armed *dynamically*, and is then called an Event Detection Point (EDP). Once a dialog between the SSF and SCF in relation to a particular call starts, the SCF (that is, the service logic processing that particular instance of the service) can request the SSF to arm EDPs for the call. When the BCP encounters an EDP it sends and appropriate Information Flow to the SCF based process as part of the dialog for that call.

---

[5] In fact, it is not any PIC. CS – 1 specifies that only some "jumps" from a DP to a PIC not corresponding to it in the BCSM are allowed. Namely, suspension of the BCP at any of the DPs 1, 2, 3, 4, 5, 6 and 9 is allowed to bring about recommencement at any one among PICs 2, 3 and 4; also, suspension at either DP 13 or 14 may lead to resumption at PIC 8.

In addition, there is another elaboration. Both TDPs and EDPs can be *requests* or *notifications*. Thus there are the following types of DPs: TDP_R, TDP_N, EDP_R and EDP_N. DPs can be armed as more than one of these types at the same time.

The sending of an IF from the SSF to the SCF at the encounter a TDP_R starts a dialog with the SCF, which is called a control relationship. During a control relationship, which lasts as long as there are outstanding EDP_Rs armed, another service cannot be invoked, even if an armed TDP_R, with all the criteria met, is encountered [Q1214].

2.6.4.2 Specialised Resource Function

The SRF provides access to specialised resources (see §2.5.4). While providing its services, the SRF is connected to the CCF/SSF and controlled by the SCF.

CS-1 supports the following specialised resources: DTMF receiver, tone/announcements generator (provides in-band information), message sender/receiver (sends/receives electronic or voice messages to/from users) and synthesised voice/speech recognition device with interactive prompting facilities [Q1214].

2.6.4.3 Service Data Function

The SDF stores and manipulates data related to IN services that is, services provided by the IN which require SCF-based processing. In the context of CS-1 IN service processing SDF communicates exclusively with the SCF. The SCF requests information or updates of data in the SDF [Q1214].

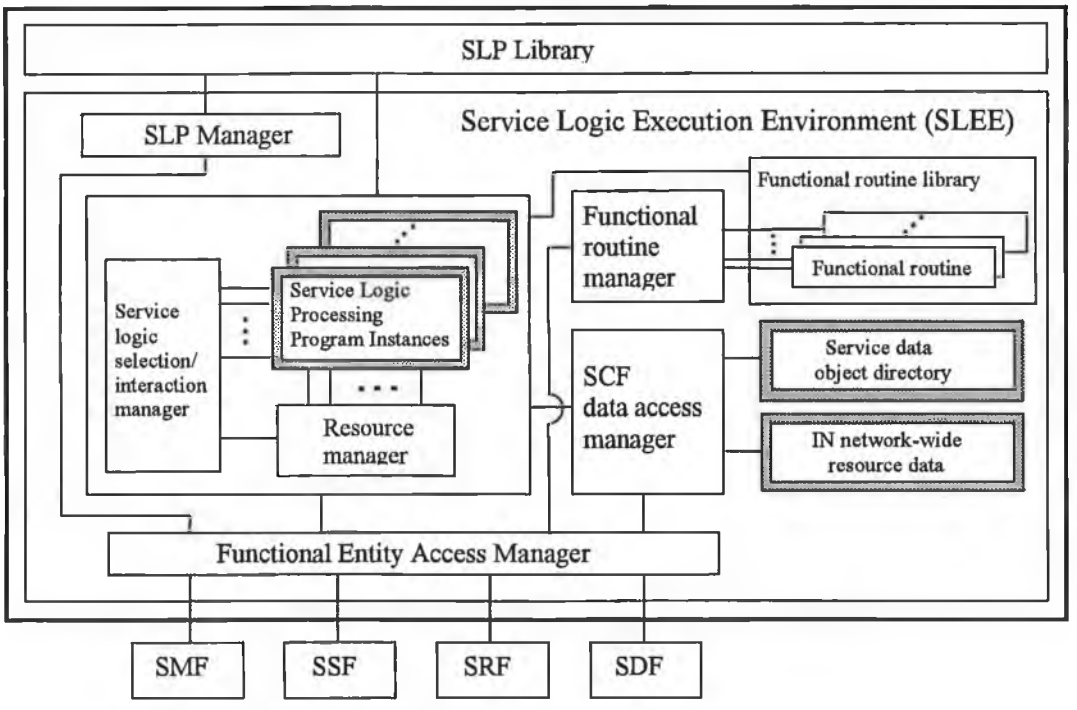
The handled data can be dynamic (changeable from the point of view of the service logic) or static (read-only from the point of view of the service logic) and they can pertain to a service feature instance (only dynamic data), a service feature or to multiple service features. Another type of data held by the SDF are references to data objects in other SDFs. The SDF returns such a reference to the SCF in the case that it does not contain the data object itself[6].

2.6.4.4 Service Control Function

Figure 2.11 shows the SCF model [Q1214]. Central to this model is the Service Logic Processing Program (SLP). The core functionality of the SCF lies in

the manipulation and execution of SLPs. A SLP is a piece of service logic, which corresponds to one IN service.

The SCF consists of the *SLP Library*, which stores the SLPs, and the *Service Logic Execution Environment (SLEE)*. The core of the SLEE is the *Service Logic Execution Manager (SLEM)*. The SLEM is assisted in the execution of SLPs by the *SCF data access manager,* which provides the functionality of accessing the SDF as well as any data stored in the SCF itself. The *Functional Routine Library* contains functional routines, which are in fact implementations of the SIBs. The *Functional Routine Manager* and *SLP Manager* manage the Functional Routines in the Functional Routine Library and SLPs in the SLP Library, respectively, as instructed by the SMF. The *Functional Entity Access Manager* provides access to the other Functional Entities. The SLEM consists of the *Service Logic Selection/Interaction Manager,* which selects and invokes *Service Logic Processing Program Instances (SLPIs)* - instances of the SLPs supplied by the SLP library - and a *Resource Manager*. The Functional Routines in the Functional Routine library are invoked by the SLPIs.



SLP – Service Logic Processing Program
SMF – Service Management Function
SSF – Service Switching Function
SRF – Specialised Resource Function
SDF – Service Data Function

Figure 2.11 – SCF Model

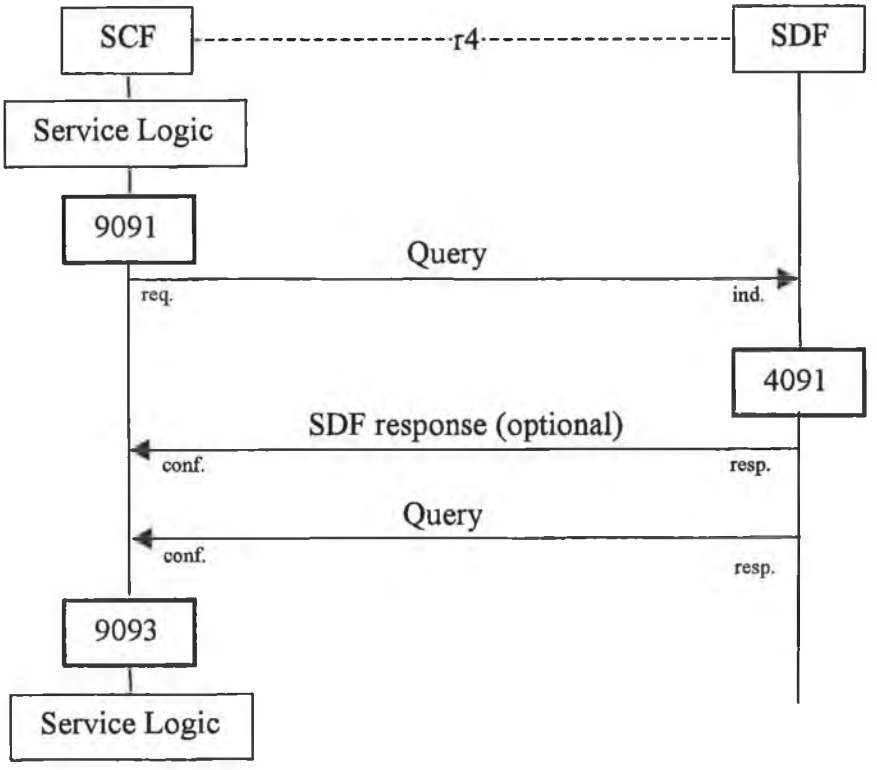
[6] The SDF to SDF query is not supported in CS-1 but is supported in CS-2.

2.6.4.5 Stage 2 SIB description

The stage 2 description of a SIB is in the form of message sequence charts, called Information Flow Diagrams, depicting the IFs between the participating FEs and the FEAs performed by them. It also includes SDL diagrams of the behaviour of those functional entities in the context of executing the SIB and textual descriptions of the IFs and FEAs used.

Figure 2.12 shows the Information Flow Diagram of the Service Data Management (SDM) SIB in the case of a query to the SDF (rather than an update request) [Q1214]. The IFs used are Query, SDF Response (Query Result is the result of Query, and is considered to be the same IF), where SDF Response is optional and can be sent to confirm the receipt of the Query, if the Query processing is going to take some time.

The Query IF has the following Information Elements (IEs): DatabaseId (optional), RequestedInformationType (optional), InformationKey (mandatory) and RequestedInfo (mandatory).



SCF – Service Control Function
SDF – Service Data Function

Figure 2.12 – Information Flow Diagram for the SDM SIB
(in the case of data retrieval)

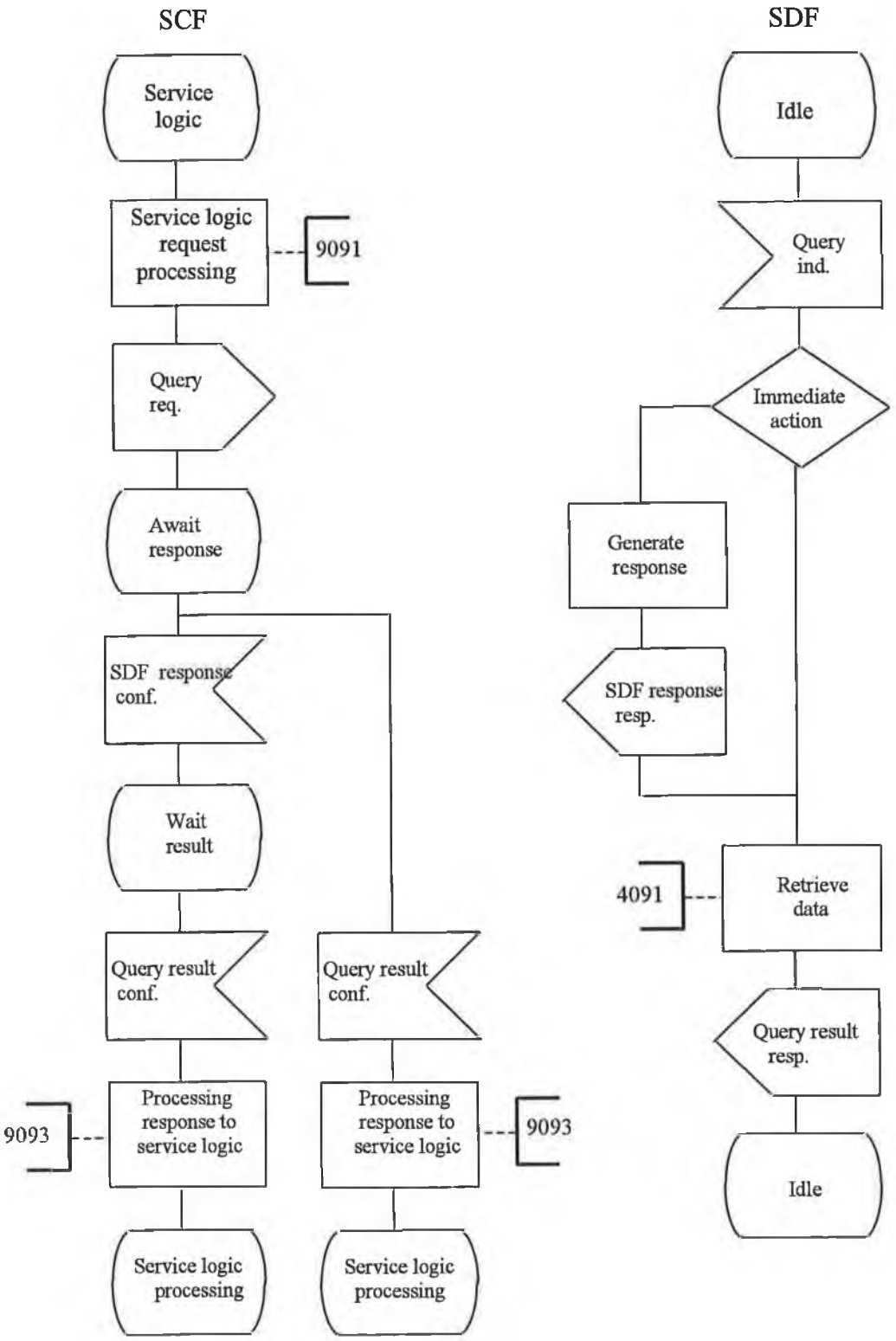


Figure 2.13 – SDL Representation of SDM SIB Functionality

The FEAs are 9091 (processes the request from service logic and generates and sends a Query_req_ind), 4091 (receives and analyses Query_req_ind, retrieves

data from database, generates and sends Query_resp_conf – the Query Result) and 9093 (receives Query_resp_conf and returns response to service logic). Here the numbers of the FEAs are in the form XYYZ, where X is the functional entity, YY is the SIB number, Z is the number of the FEA that distinguishes it from other FEAs pertaining to the same SIB and the same FE.

The SDL diagrams for this SIB are given in Figure 2.13.

It should be noted that the full description of this SIB should include the description of the update request to the SDF. However, part of the SIB description already shown is enough to illustrate the stage 2 SIB description methodology.

2.6.4.6 FE Relationships

The three relationships between FEs which have been fully described in CS-1 are: SCF-SSF, SCF-SRF and SCF-SDF. Each of these relationships is defined by the complete set of IFs which can be sent between the two involved FEs. The IFs can consist of a request/response pair or of a request only.

Some IFs are sent within SIBs, some of them are sent independently, particularly those invoking IN service processing and sent from the SSF to the SCF.

2.6.5 Physical Plane

The mapping of the CS-1 Functional Entities onto Physical Entities is the same as that described in the general section about the Physical Plane (see §2.5.5) [Q1215].

For communication on the SCF[7]-SSF and SCF-SRF interfaces, CS-1 IN uses the Signalling System No. 7 (SS7) protocol stack[8] with ROSE[9] based Transaction Capabilities Application Part (TCAP) to invoke IN-specific Intelligent Network Application Part (INAP) operations. For the SCF-SDF interface SS7 TCAP is also used, however, the operations lie on top of a different protocol layer, the ITU-T-defined Directory Protocol [Q1218].

The communication between the SSF and the SRF is through the ISDN Basic Rate Interface or Primary Rate Interface. If the SSP (containing the SSF) is to act as a

---

[7] The protocols, even though they are physically used on connections between PEs, are really connecting FEs located in the PEs.

[8] The lower layers are the Message Transfer Part (MTP) and the Signalling Connection Control Part (SCCP).

[9] ROSE stands for Remote Operations Service Element and as a protocol sub - layer of TCAP provides remote operations functionality. The primitives provided by ROSE are: Invoke, Return Result, Return Error and Reject.

relay between the SCF and SRF, it uses the D-channel of the ISDN connection for signalling.

The mapping of Information Flows of the DFP onto INAP operations is almost on a one-to-one basis [Q1218].

2.7 Capability Set 2 and Beyond

2.7.1 Capability Set 2

The general capabilities included in CS-2, apart from those already in CS-1, are: a limited set of mobility and broadband ISDN functions, service customisation and call party handling (CPH), which, being relevant to this project, is described in more detail in a separate section below. Also, there is more attention given to the management and service creation functions in the network, although no interfaces have been specified in that area [Q1221].

The view on the Global Functional Plane reveals another basic process, the Basic Call-Unrelated Process, which models capabilities not associated with a particular call. Such capabilities are, for example, user registration, activation or deactivation (used in Personal Communication Services – PCS). The SIB concept is changed to denote a library of related instructions, called SIB operations, rather than an atomic action. High level SIBs are introduced, to accommodate frequently used and repeatable SIB chains, as well as parallel processing of more than one SIB chain, and subsequently inter-process communication between SIB-chains. New SIBs have been introduced for the purposes of call party handling [Q1223].

CS-2 introduces some new functional entities to support mobile communications functionality and the call-unrelated activities of the end-user. For call-unrelated activities those are the Service Control User Access Function (SCUAF - equivalent to the CCAF) and the Call-Unrelated Service Function (CUSF – equivalent to the CCF/SSF). The BCSM is considerably changed, with more PICs and DPs. Standard call-unrelated activities are modelled in the Basic Call-Unrelated State Model (BCUSM).

In general, CS-2 has a wider scope than CS-1 (includes mobile communication functionality), tackles a larger part of the IN (management and service creation are included), widens the set of services catered for (customisation

by the user, multiple-party call handling) and offers more elaborate "tools" for the design and creation of services (detailed BCSM and complex SIBs).

The CS-2 INAP is a superset of CS-1 INAP, extended to accommodate the additional capabilities of CS-2 [Q1228].

2.7.1.1 Call Party Handling

Within the IN context, Call Party Handling (CPH) denotes the capabilities of the SSF/CCF to manipulate calls, that is, parties involved in calls, beyond the basic two-way call in which one party is the caller and the other is called. Such manipulation includes call parties being put on hold, call transfers, multiple-party calls and calls initiated from the network (rather than by an end user).

The CPH of CS-2, as described in [Q1224] relies on a well-defined set of basic capabilities of the switching system (such as the capability to detect a hook-flash and act on it). The scope of the CS-2 CPH is defined through a number of call states (e.g. "Call on Hold") and state transitions, which also provide the SCF with a view of the call. Two approaches to providing the CS-2 CPH capabilities are recommended. One is for the SSF to provide a comprehensive group of capabilities, described through a group of call states called Connection View States. The other approach, called the Hybrid Approach is for the SSF to provide a limited set of capabilities, contained in a smaller number of states, with the remainder of the capabilities being achieved in conjunction with the SRF, by using bridges etc.

2.7.2 CS-3 and Future Capability Sets

Work on Capability Set 3 is still under way. The CS-3 objectives are to develop a protocol which will support Broadband ISDN and Wireless Personal Communications Services, to standardise service creation and to apply the Telecommunication Management Network (TMN[10]) to IN, as the standard for IN management.

The IN, however, is designed to evolve much further, towards the IN target architecture [Q1201], which could be based on the results of TINA, described below [Fayn96].

---

[10] TMN is an Open Systems Interconnection (OSI) – based, ITU – T developed, standard specifying interfaces between managed and managing entities in telecommunication networks, used in providing the Operations, Administration, Maintenance, and Provisioning (OAM&P) functionality in those networks.

2.7.3 TINA

TINA stands for Telecommunications Information Networking Architecture and is a project undertaken by the TINA Consortium (TINA-C). The Consortium was formed in 1993 and has many members including network operators, telecommunication equipment manufacturers and computer manufacturers.

The objective of the TINA project is to define an all-encompassing (from the aspect of network technologies, processing technologies and services alike) architecture for the future telecommunication network, with an accent on broadband communications, multimedia and multiple-party services and distributed processing.

TINA recognises and addresses three aspects of service provisioning and operation in the telecommunication network and, accordingly, singles out three architecture facets to be defined: a Computing Architecture, a Management Architecture and a Service Architecture.

The Computing Architecture deals with the details of the Distributed Processing Environment (DPE) and is partly based on the work of the Object Management Group (OMG) and its Common Object Request Broker Architecture (CORBA) standards. The Management Architecture deals with all the traditional types of management in networks, as well as with Connection Management, which is the provision of the actual end-to-end connections. The Service Architecture describes the components which take part in service provision [Dupu95].

2.8 Conclusion

This chapter has dealt with the Intelligent Network and the methods used for its definition and description. A good understanding of the structure and functionality of IN is vital for its optimal utilisation in designing a system that extends it.

The IN capabilities, in the form of services, service features, and especially SIBs, represent functionality that should be highly reusable in a system extending the IN to include the Internet. The concept of the Basic Call State Model and Detection Points, particularly as they are already in place in existing networks, would ease the design and incorporation of services into the extended system in cases where new functionality needs to be introduced. All this, indeed, is a consequence of the fact that services offered by the extended system are still in part (that being the part that operates in the telephone network) IN services by nature.

The capabilities superimposed on the IN to form the extended system shall be described by the methods used for IN specification. Therefore views of the system on each plane of the Intelligent Network Conceptual Model shall be given, as they have been presented in this chapter and can be found in the ITU-T Recommendations of the Q-series.

3 Telephone Network – Computer Network Integration

3.1 Computer Networks

3.1.1 Introduction

Today, computer networks are a diverse and ubiquitous phenomenon. Great diversity exists among both technologies applied to their implementation and the applications that make use of them. The first incentive for the development of computer networks was the sharing of processing resources between users. But ever since their invention, they have been used for much more than that, including the sharing of information and resources other than processors, communication between users in various forms and distributed processing.

Owing to the packet-switched nature of communication on computer networks, which is the optimal technology in the conditions of bursty traffic (characteristic of conventional applications in computer networks), they have not been used for communication involving real-time information flows, such as voice communication. Recently, however, that has also changed – the ever higher bandwidths of new transmission technologies and improved protocols have made it possible – and there are various finished products that offer voice communication over computer networks.

Generally speaking, there are two types of computer networks: Local Area Networks (LANs) and Wide Area Networks (WANs). LANs are privately owned and used by single organisations and usually cover an area of up to 10 km in diameter. WANs span a much larger geographic area and offer services to the public. These two types of networks also differ in the transmission technologies they use. LANs use very economical technologies, which, however, have restrictions as to the size of network they can comprise. WAN technologies, on the other hand are more expensive, but allow for large networks. Another type of network is a Metropolitan Area Network (MAN). From the technological point of view, MANs are similar to LANs, except that they allow for higher bandwidths and can cover larger geographic areas, of over 100 kilometres [Stam94][Come97].

While WANs and LANs employ different technologies for the delivery of information, they can be interconnected in order to communicate on a higher level, through common protocols. The Internet is a world-wide network of interconnected, physically heterogeneous WANs and LANs, with all the computers attached to them sharing the same address space and communicating by the same protocol.

3.1.2 Short History

As a matter of interest, the first transmission ever of computer data between two sites made use of a telegraph connection. George Stibitz, of Bell Labs, demonstrated the Complex Calculator[11], located in New York, to attendees at an American Mathematical Society meeting, gathered in New Hampshire, by accessing the calculator through a terminal. This was in 1940.

The first paper on packet switching theory, by Leonard Kleinrock, of MIT, was published in 1961. He described packet switching and its feasibility, in contrast to circuits, for communications between computers. The first packet switched network, ARPANET (after the Defense Advance Research Projects Agency - DARPA) was completed in the USA at the end of 1970. It could perform file transfer and remote log-in. Email was available on it by July 1971 [Salu95].

Around the same time, in 1970, the first packet radio network (ALOHA) was built between sites of the University of Hawaii (on different islands). A similar idea was applied to a coaxial cable and Ethernet, the most common LAN technology, was born - in 1973, in the Palo Alto Research Centre.

The TCP/IP protocol was developed to solve the problems of reliability, interconnection of networks and a need for big numbers of nodes, all present in ARPANET. The "switch" to TCP/IP on ARPANET and its related networks took place on January 1st, 1983. Also, other networks such as USENET, Fidonet and BITNET were formed in the late seventies and early eighties.

TCP/IP got its "big break" in 1986, when it was chosen as the protocol to be used for NSFNET (the National Science Foundation – NSF) in the USA. NSFNET served as the backbone connecting numerous regional networks, soon many of them outside the USA [Salu95].

The first WWW software was developed by Tim Berners-Lee, of CERN (the European Laboratory for Particle Physics), in 1990, with the aim of providing a

[11] An electromechanical device, built from telecommunication components - relays and crossbar switches, which could perform calculations with complex numbers.

practical way for high-energy physicists around the world to share information. However, it quickly spread into the wider community of Internet users, becoming the WWW and having a profound effect on the way we view and use the Internet [Bern96].

3.2 The Internet

Intuitively, most people would describe the Internet as a heterogeneous, world-wide network of computers, consisting of several smaller networks, and connecting millions of users. However, that still does not entirely define it. The name Internet is, in fact, reserved for such a network, of interconnected WANs, LANs and single computers, in which each computer has a unique address complying to a certain global addressing scheme.

3.2.1 The Basics

3.2.1.1 TCP/IP

The first computer networks were small and used simple protocols which were not very reliable and supported limited network sizes. As these networks grew, and the need arose for their interconnection, a project was started, by several co-operating organisations in the USA in the 1970s, to devise a new protocol, which would serve all the needs of a complex and reliable network. The result was the Transmission Control Protocol/Internet Protocol (TCP/IP), which, in fact, consists of two protocols: IP, an unreliable end-to-end packet delivery protocol, and TCP, a reliable, connection oriented transport protocol, which uses IP [Come97].

The above-mentioned global addressing scheme is the scheme devised as the most important feature of IP. The currently applied version of IP uses 32 bit addresses, which are usually represented as decimal integers (each for one byte), separated by dots (e.g. 136.206.36.139). This format is called the *dotted decimal notation* and the type of address itself is called an *IP address*.

TCP is not the only protocol used on the Internet on top of IP, but it is the first invented and the most frequently used. It provides reliable while fastest possible transport by employing several techniques such as retransmission, congestion control and "window" flow control. TCP is a connection-based protocol (users must establish a logical connection before exchanging information). Its connectionless counterpart is the User Datagram Protocol (UDP).

The currently used version of IP is IPv4. However, a new version called IPng (IP – The Next Generation) or IPv6, is being designed and is currently an Internet Engineering Task Force (IETF) draft standard. IPv6, among other things, uses 128-bit addresses (to accommodate the growth of the Internet for many years to come), provides support for audio and video (by associating packets with particular, high quality paths) and has been designed to be extensible, unlike its predecessor, IPv4 [Come97].

3.2.1.2 The Domain Naming System

The dotted decimal notation is more legible to humans than the hexadecimal representation of the Internet address, however, there is an even more human-friendly way of addressing computers on the Internet, called the Domain Naming System (DNS). In DNS, computers are represented as a sequence of words (with or without meaning) separated by dots, for example: *computername.subdomainname.companyname.com.* The word at the very right denotes the top-level domain (e.g. com for companies, edu for educational institutions; or by country: ie for Ireland, yu for Yugoslavia). The words to the left of the top-level domain name are names of lower level domains or subdomains. The leftmost word is the name of the computer. A DNS name can be used instead of an IP address by an application. Before any communication is attempted, the name has to be *resolved* (translated into an IP address). This is done by a name resolver, which may have to communicate with different servers in the Internet in order to get the address [RFC1883].

3.2.2 Internet Applications and High-Level Protocols

3.2.2.1 Applications

Any application, which employs computer communications using TCP/IP (or some other protocol combination with IP), is an Internet application. The oldest and simplest applications are Telnet (for **Tel**ecommunication **Net**work) and FTP (File Transfer Protocol). Telnet allows a person using one computer to log-in, by using a username and password, into another computer and thus gain access to that other computers storage and processing power, emulating a terminal. FTP allows files to be transferred between computers.

Another early Internet application, and the first one providing communication between people, is electronic mail, or email. Email gives the user the possibility to send messages to "mailboxes" on other computers, belonging to other users, which those other users can then access in order to read the messages. Email is still one of the most widely used Internet applications.

The other most widespread Internet application, invented more recently, is the World Wide Web (WWW). The WWW is a distributed system of interconnected documents and other types of information with a mechanism that allows the WWW user to access that information in an interactive manner. As one of the most important Internet applications and one heavily utilised in this project, the WWW is described in some more detail in a separate section below.

3.2.2.2 High Level Protocols

High-level protocols are protocols that have application-specific semantics and use a transport protocol to take care of delivering raw data from originator to destination. If a high-level protocol is used for communication on the Internet, the transport protocol that it uses is TCP, or in some cases UDP. Here such protocols most commonly used on the Internet are described.

The TELNET protocol provides bi-directional character-based communication. It uses a group of control characters (which are transferred between the communicating parties alongside characters carrying data) and a set of rules for the behaviour of the communicating parties to offer a simple standard means of communication between two entities, most commonly between an emulated terminal and a host [RFC854].

Any File Transfer Protocol (FTP) application, as described above, is in fact based on the protocol that gives it its name. FTP uses two connections between the communicating parties, a TELNET-based control connection and a connection for the transport of raw data. The control connection carries commands for setting options pertaining to data transfer (e.g. whether the data is in text or binary format) and for requesting the data transfer [RFC959].

The Simple Mail Transfer Protocol (SMTP) is the protocol used for sending email between hosts on the Internet. It is a text-based protocol, which makes all the commands and data exchanged between the communicating parties understandable to humans at a first glance [RFC821].

The Hypertext Transfer Protocol (HTTP) is the protocol used for transactions on the WWW. Those transactions consist of transfer of requests and document data. The HTTP commands and responses are issued in text format, the same as in SMTP [RFC2068]. Therefore, both HTTP and SMTP commands can be transported between the communicating parties over TELNET.

3.2.2.3 The World Wide Web

The World Wide Web (WWW) is an Internet application, probably the most popular one, besides email. The WWW is exactly what its name suggests, a world-wide web – of hypertext documents.

The concept of hypertext, as documents interconnected into a web through links and consequently easily accessible, existed for some time before the appearance of the WWW itself. A hypertext link is some kind of an indicator, contained in a document, of the location of another document, which allows any viewer of the first document to access the other document.

The two types of functional entities associated with the WWW are *WWW servers* and *browsers*. A WWW server is a server program, which can receive requests for documents, finds each requested document, and send it to the computer that issued the request for it. Each WWW server handles documents located on the same computer as itself. A browser is a client application, which can issue requests for documents, receive the documents and display them. A user uses the browser to access documents on a server directly or through a link from some other document, which may be on another server.

The WWW uses documents written in the Hypertext Markup Language (HTML) [RFC1866]. Apart from providing a mechanism for specifying links to other documents, HTML allows the user to apply complex formatting to the text and to incorporate various things, such as graphics, images, video and sound, into the document. The links in HTML are simply graphics or text, which are highlighted in some way and can be clicked on with the mouse. A mouse-click on a hypertext link invokes a request for a new document, from any server.

The protocol used for transferring html files (files containing HTML-specified documents) is called Hypertext Transfer Protocol (HTTP). The browser issues requests for documents by using a Uniform Resource Locator (URL), which consists

of the name of the protocol to be used[12], the DNS name of the computer where the document is located and the name of the document (for example: *http://www.teltec.dcu.ie/info.html*) [RFC1738].

A URL, however, does not have to be associated with a static document (HTML file). It could also be associated with a script that creates documents "on the fly". Such scripts are called Common Gateway Interface (CGI) scripts, and are most often written in Perl or C [Bout96]. The name, CGI, comes from the fact that these scripts can provide the WWW server with an interface to "back end" programmes, such as databases. An important role of the CGI script is form processing. HTML includes means of specifying some standard form elements, such as text fields, radio buttons, check boxes and selection menus. The data contained in an HTML form is conveyed to the WWW server, and through the server to the CGI program, once the form is submitted by the browser as an HTTP request. The CGI program can then process the request and form an appropriate WWW page to be returned to the browser.

As if this were not enough, the HTML documents, which can be accessed on the WWW, need not be static, whether already existent or created "on the fly". There are two older techniques used for creating dynamic documents, descriptively called the browser pull and server push techniques. With browser pull, the browser loading the document is specially instructed to reload the document after a certain amount of time. If this is done each time a document is loaded into the browser, the document will continuously be reloaded until the user decides to move to a different URL. If the document is created with a CGI script which produces a different document every time it is called or the document is changed in time in some other way, what appears to the user is a changing, that is, a dynamic document, e.g. a slide show. The other technique, server push, breaks the rule by which one URL produces only one HTML page at a time. With server push, a special indication is given to the browser that it is being sent several pieces of HTML, which are to be loaded one by one, and one over the other. The effect is similar to that got with browser pull, except that server push allows for only a part of the HTML document to be dynamic, while browser pull obliges the browser to reload the whole HTML page.

More recent developments in dynamic documents are so-called active documents, which include code that is to be executed by the browser. Most browsers today have the in-built capability to run such programs. The programs may

[12] Browsers support protocols other than http, for example ftp.

dynamically affect the appearance of the document in the browser or conduct user interaction. Java is the most popular method for constructing active documents. Java is a half-compiled, half-interpreted programming language [Flan97]. It provides a method for creating special applications (applets) which can be embedded into an HTML document. In order to run applets the browser must contain an execution environment, in fact an interface between the operating system and the Java program, to provide universal portability and security, among other things. Java, although the most popular, is not the only method for producing dynamic documents. To mention but one other, Javascript is a scripting language used for similar purposes.

3.2.3 Internet Application Implementation

3.2.3.1 The Client-Server Model

While the first two of the above-described applications, Telnet and FTP, as well as the WWW allow the user to access a remote site and to use resources or get information from that site, email allows him/her to send information to the remote site. This difference in the communication dynamics does exist at the level of human perception. However, at the level of computer application communication any dialogue between two computers can be described in terms of the simple *client-server model.*

In order for any type of communication between two computers to start, the computer initiating the communication has to send, asynchronously[13], some initial information to the other computer and there has to be an entity in the receiving computer, which can understand this initial information (at the application level). This entity is alerted when the initial information arrives (as a telephone rings to alert a person with whom someone at another phone wishes to communicate) and then takes appropriate action, whether it be only processing or it also includes further communication with the initiator. Such an entity must be active (running) throughout the time while someone might require to communicate with it, and it is called a server. The initiator of the communication is called the client. Accordingly, if someone wishes to access files on another computer using FTP, the remote computer must have an active FTP server program, which handles requests from the client. If

[13] Asynchronously, in the sense that there is no way of predicting the event. It is either caused by human action or by some entirely independent (from the receiving computer's activities) activities of the sending computer. Therefore, the sound coming from an alarm clock arriving at the ear of the person who set it, is not asynchronous in this sense, even if it was set a " very long" time ago.

an email is sent, the receiving system must have an active mail server program, which will accept the message and "put it into" the right mailbox. So-called peer-to-peer communication is a special case of client-server communication, where both entities play the role of both client and server at the same time. Finally, any WWW transaction consists of a request to a WWW server and a response containing the requested information, usually an HTML document.

3.2.3.2 Programming Methods

3.2.3.2.1 Sockets

Sockets represent the simplest and oldest application-level implementation of the client-server model. Sockets are, basically, an Application Programming Interface (API). The function calls contained in this API abstract the transport protocol (most often TCP) into the group of communication activities controlled by the application. Those are: specification of protocol to be used (TCP, UDP or other), specification of remote address (client), listening for initial communication from a client (server), initiation of communication (client), sending information (client and server) and receiving information (client and server).

3.2.3.2.2 CORBA

Common Object Request Broker Architecture (CORBA) is the Object Management Group's (OMG) standard for defining objects and communication mechanisms in heterogeneous distributed computing environments [OMG95]. The OMG has standardised the Interface Definition Language (IDL) for defining interfaces on the objects of a distributed system, as well as standard interfaces, defined in IDL, for necessary or useful mechanisms such as locating objects in such a system.

The various CORBA implementations (supplied by CORBA vendors) provide an automatic method for adding network communication functionality to the application functionality of a system, given the IDL descriptions of the interfaces that are to be present in the system. At the same time, CORBA implementations provide the functionality behind the standard OMG interfaces.

If an interface described in IDL contains function calls (an interface may describe solely data types), the object implementing these functions is called a server for that interface, while the object making the function calls is a client for that

interface. The function calls made in the client are converted into messages and sent to the server over some transmission medium. The return value of a function call (if there is one) is also converted into a message and sent back to the client. The reason CORBA is interesting in the context of communication on the Internet is that the transmission of these messages can be (and mostly is) undertaken over TCP/IP. The code for the conversion, between function calls/function return data on the one side and messages that can be sent "over the wire" on the other side, is generated by a CORBA "compiler" which uses IDL as input. This is the automatic method mentioned earlier, for incorporating the communication software into the system, which is provided by the CORBA vendors. The overall communication functionality comprises an abstract "communication bus", called the Object Request Broker (ORB), accessible to all the objects in a CORBA system.

The OMG facilitates inter-operability between ORBs provided by different vendors through a standardised protocol for communication between CORBA objects, the Internet Inter-Orb Protocol (IIOP [14]).

CORBA makes the implementation of distributed systems independent of the final location and distribution of the system's constituent objects. It also makes implementation language independent by allowing different objects in the same system to be implemented in different programming languages, of which C++ and Java are most commonly used. Using CORBA also spares the programmer from having to implement the details of network communication.

It should be noted that using Java in conjunction with CORBA makes for "portable clients". A CORBA-enabled Java Applet can be downloaded as part of an HTML page and can then, as a client, access a CORBA server on the Internet, which does not have to be at the same location as the WWW server containing the downloaded HTML page.

3.2.4 Intranets and Extranets

Internet technology as described above may also be used within organisations or companies in their private computer networks so that WWW browsers, email and newsgroup readers are used for all internal communication and for sharing of information such as internal telephone directories and manuals. Such a network, which may or may not be open to the Internet, but is not accessible from the outside, is called an Intranet.

An Extranet is the same as an Intranet, except that access to it is selectively allowed to other organisations or companies allied to the owner of the Extranet, such as suppliers or customers.

3.3 Computer-Telephony Integration

3.3.1 Introduction

Co-operation between telecommunication and computer systems is not a new phenomenon. Computer technology was applied in telecommunications first in stored-program switches and for the processing of billing information and later in Intelligent Networks, to provide various services. At the same time, telecommunication channels were used to build computer networks. However, true interaction between the telecommunication and the computer world, from the point of view of the user, occurs where a computer at the user's premises interacts with the user's telephone to provide new functionality, through heavy use of user-tailored resources and user-specific information. Computer Telephony Integration (CTI) denotes the principle of integrating computer and telephone functionality at the user-resource level, and Computer Telephony (CT) denotes the result of that integration [Stra96].

The first applications based on CTI appeared in the early 1980s, around the same time as the first Intelligent Network, but were used only in large call centres, where building of such systems was viable thanks to the large call volumes [Udel94]. Call centres employ applications of which Automatic Call Distribution (ACD) is the most typical example. ACD performs routing of incoming calls directly to free agents and, as an agent accepts the call, sends him/her information about the calling customer, in the form of an automatic pop-up on their screen. At the time when they first appeared, such systems would have been based in a central computer (mainframe or minicomputer), which would have had a connection to the call centre's PBX as well as connections to all the agents' terminals.

Truly fertile ground for the growth of CTI was cleared with the proliferation of desktop intelligence, namely, the use of PCs. Today, a LAN telephony server has replaced the minicomputer of the early CT systems and PCs connected to the LAN

[14] IIOP uses the TCP/IP protocol combination for transport.

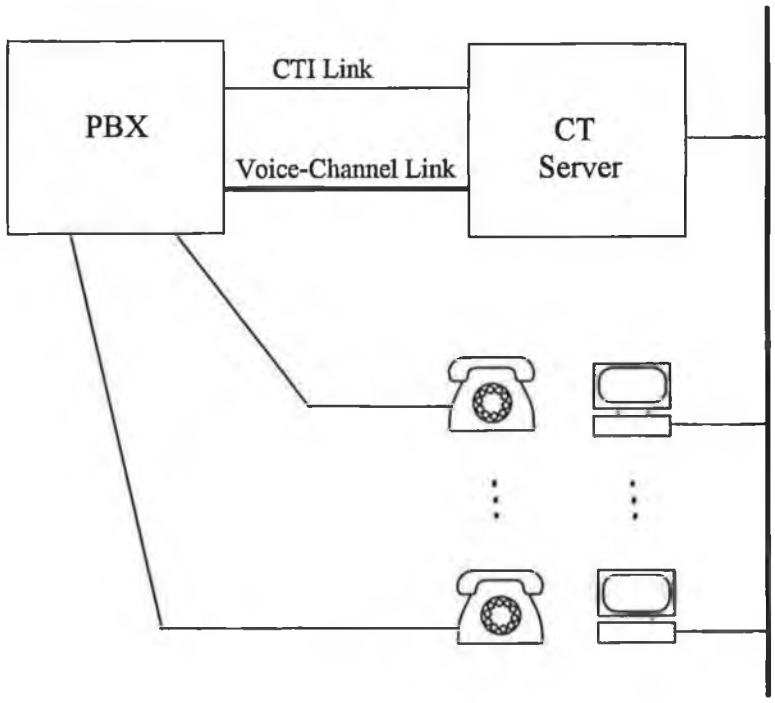
are used instead of the terminals. The LAN telephony server exerts control over the telephone part of the system on behalf of the users connected to the LAN. Simultaneously, there is another approach to CTI, whereby the CT application is based in one user's PC and controls exclusively that user's telephone line. From a business point of view, the first approach suits companies or organisations big enough to have a LAN installed on their premises, whereas the latter is more appropriate for very small businesses and work-from-home arrangements [Titt96].

3.3.2 CTI Description

A CTI system may be based on shared resources or resources that are not shared [Stra96]. A LAN–based CT system (shown in Figure 3.1) employs shared hardware and software resources, located on the telephony server. Shared software resources include, among others, PBX-communication drivers, call control software, application software and information databases. These resources use the CTI link to communicate with the PBX. Shared hardware resources usually consist of media-processing PC cards, such as voice digitisers, facsimile cards, Interactive Voice Response (IVR) units etc., plugged into a time-division type of bus, such as the MVIP bus or Scbus [Fleg96], devised specially to accommodate digitised voice-path signals between PC cards. The hardware resources in the CT server exchange information with the PBX through a voice-channel link. If the resources are not shared, they are based on one user's PC and they are used with the telephone line belonging to that user. These resources are applications, call control software (different from that employed by a LAN telephony server), and various media processing PC cards, very often combining multiple functionality, such as modem, facsimile, voice messaging and type-of-signal recognition.

All CTI applications entail some kind of control of the telephone part of the system by the computer part of the system. Although stand-alone PBXs already provide many such features, Computer Telephony applications extend those processing capabilities dramatically and in several ways. First, a CT application may provide a computer-based user-interface and with it introduce entirely new functionality. For example, ACD provides automatic pop-ups of customer information at agents' computer screens. On the other hand, CT may replicate features already offered by PBXs, while simplifying and encouraging their use with the much more user-friendly computer-based interface: a click of the mouse to invoke, for example, a call-back, replaces the dialling of special key sequences on the

phone. Second, CTI allows extensive use of user-specific information and resources. For example, an ACD application uses a customer-information database, specific to the ACD business user from which information for the pop-ups is drawn. The Fax-on-Demand (FOD) application uses a user-specific library of facsimile documents, which are faxed out to customers who request them through automated interaction with the application. Third, CT allows for unifying, where applicable, computer and telecommunication functionality. This is what unified messaging applications do, by receiving and organising voice mail, email and facsimile messages all in the same "mailbox". Finally, existing CT APIs (described in §3.3.5) allow for the development of various user-specific CT applications.



PBX – Private Branch Exchange
CT Server – Computer Telephony Server
CTI – Computer Telephony Integration

Figure 3.1 – A Computer Telephony System with Shared Resources and Third Party Call-Control

The control that the computer part of the system exercises over the telephone part of the system has two functional aspects: call control and media control [ECMA269]. Call control entails manipulations of connections and parties (or devices) involved in a telephone call. An example of call control is the routing of a call depending on the caller's number, or an ACD application, already mentioned

above, or the connection of an incoming call to a messaging system if the called party is busy. Media control involves manipulations of devices, which provide media streams to be sent over the telephone line or are able to interpret and process such streams. An example is an Interactive Voice Response (IVR) unit, which can run entire scripts consisting of message playing, DTMF detection, voice recognition etc., or a computer facsimile card. The majority of CT applications involve the combination of these two types of control.

Depending on how it is carried out, call control falls into one of two categories. When the connections of a telephone line are controlled by an application on behalf of the communicating entity, be it the user or another application, such as a facsimile-sending application, what is taking place is first-party call control [Stra94]. First party call control utilises standard terminal equipment signalling, such as off-hook, on-hook and DTMF digits (for POTS) or other, depending on the technology (e.g. ISDN layer 3 signals). In third-party control, the controlling entity is an application, which does not have access to the communication path of the controlled lines. Third party call control requires a special signalling link, called a CTI link, between the controlling application and the switching entity, usually a PBX. CTI links can be implemented using either a proprietary protocol (e.g. Northern Telecom's Meridian Link Protocol) or a standard protocol, such as the ECMA Computer Supported Telecommunication Applications (CSTA) protocol [ECMA217][ECMA269]. An alternative, although rarely used, to the CTI protocol is an SS7 connection from the user-premises computer to the public telephone network. Sometimes long-distance carriers offer such connections, however, via appropriate security installations, such as firewalls [Stra96].

In most cases shared-resources systems rely on third-party call control, while non-resource-sharing systems most often employ first-party call control.

3.3.3 CTI Standardisation

As the number of players in the computer industry grows and computer systems tend towards distribution, industry standards are needed in order to facilitate the integration of the multiplicity of heterogeneous products and enhance their marketability. The same stands, specifically, for Computer Telephony systems. There are several vendor groups and organisations working on the standardisation of CTI. Some of the most active of those organisations are Enterprise Computer Telephony

Forum (ECTF)[ECTF], Versit, and ECMA (formerly European Computer Manufacturers Association).

ECMA is an international standards body, which aims to produce standards for certain aspects of information technology and telecommunications. ECMA's Task Group Eleven (TG11) is developing the Computer Supported Telephony Application (CSTA) standards, which pertain to CTI. The work on CSTA started in 1988, and has been undertaken in phases, the phase currently under development being Phase III. Some of the changes introduced in Phase III are based on contributions from other organisations such as the ECTF and Versit. In general, CSTA standards are evolving towards more structured and elaborate specifications of existing material as well as towards encompassing additional aspects of CT functionality, the standardisation of which may be crucial for higher interoperability.

3.3.3.1 Overview of CSTA

CSTA is a standard for building third-party call control CTI systems with shared resources and it is here described as representative of CTI systems in general [Ansc96]. CSTA views its area of operation as consisting of three domains: the switching, the computing and the special resource domains [ECMA217]. The domain of operation of a CSTA application spans parts of at least two different domains, as shown in Figure 3.2.

The behaviour of terminal devices in the switching domain is described in a Connection State model, analogous to the BCSM of the Intelligent Network. A Call State is defined as a sum of the Connection states of all the devices involved in the call and these are analogous to the Call View States offered within the context of CPH in IN. Analogy can also be seen in the division of the overall CSTA domain into the tree domains and the fact that most of the control in CSTA is exercised on the part of the computing domain over the switching and special resource domains. A group of event reports, mostly sent from the switching domain to the computing domain are defined and an analogy can be seen here with the BCSM Detection Points in the IN.

3.3.4 The Telephony APIs

Telephony APIs are sets of functions, which provide the Computer Telephony application programmer with access to the basic functionality of a CT system [McCo97]. Several APIs have been defined and implemented by various players in the CT industry. The most popular ones are Microsoft's TAPI (Telephony

Application Programming Interface), TSAPI (Telephony Services Application Programming Interface) – specified by AT&T and Novell and JTAPI (Java Telephony Application Programming Interface) – specified by the Sun Microsystems.

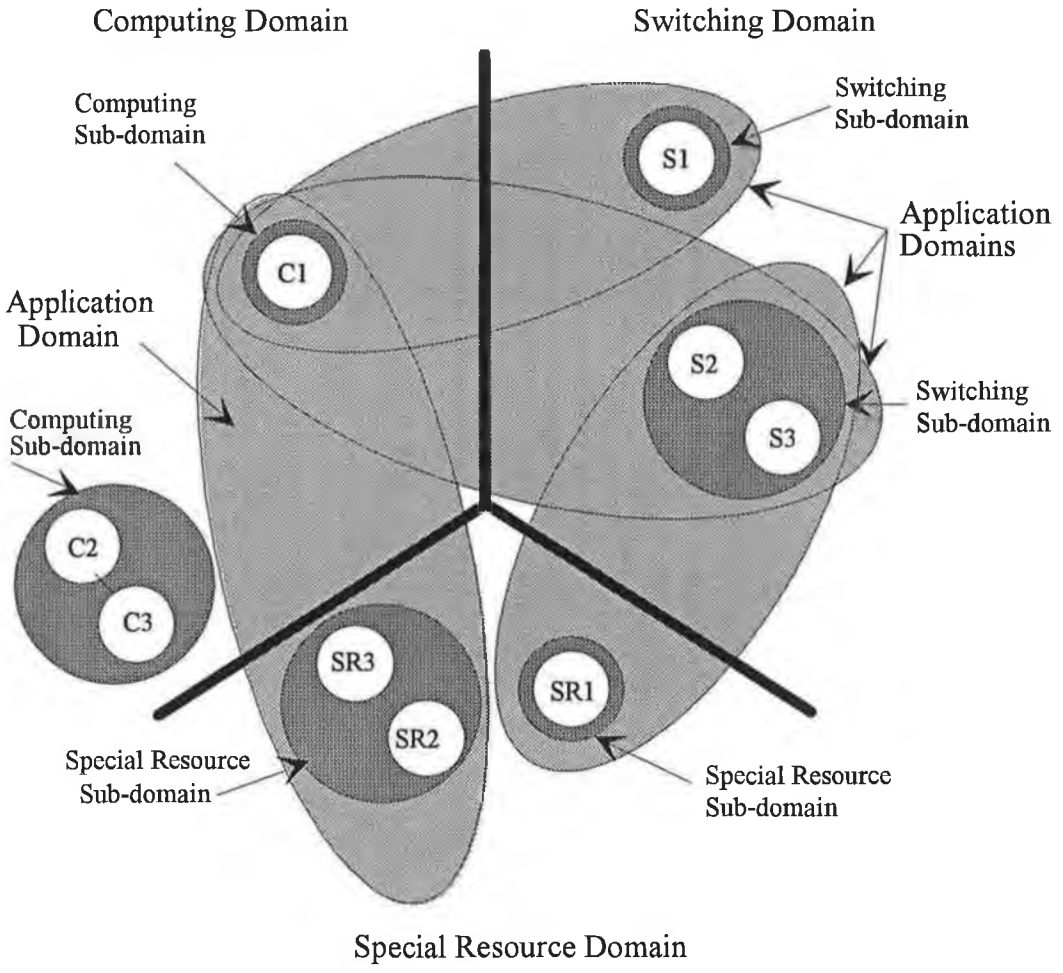


Figure 3.2 – CSTA Domains and Sub-Domains

TSAPI is based on CSTA [Cron96] and is and API for building applications based on third-party control and shared resources. TAPI was originally designed to provide an interface to first-party call control functions, for Windows based CT application programmers, but the latest version of TAPI, namely TAPI 2.1, implements a third-party call control system interface. JTAPI is envisaged to offer either an interface to CT functionality through other APIs, as an additional layer of software above, for example, TAPI or TSAPI, or a "direct" interface to CT hardware.

3.4 Internet-Telephony Integration

3.4.1 Introduction

The integration of the Internet and telephony, in any of its forms (public or private, wired or mobile), is a natural consequence of the rapid and all-encompassing growth of the Internet and its permeation of almost every area of human activity. However, there is a parallelism between the PC and the telephone as the two most ubiquitously used personal devices and mutual complementing between their functions, which further emphasises the possibility for such integration to become just as widespread and significant as its constituent parts. That having been said, and although work has been taking place towards Internet-telephony integration within various organisations and companies, the actual implementations have, up till now, been sporadic and diverse in form and standardisation is still only in its early stages [PINT99].

The integration of telephony and the Internet, as it will be discussed here, is the provision for inter-working between some functional aspects of a telephone network and some functional aspects of the Internet in order to create new services/applications. Therefore, for example voice-over-IP is a purely Internet application, whilst a mechanism which allows a voice call between an Internet phone and a POTS phone falls under Internet-telephony integration.

Internet-telephony integration denotes the integration between the Internet and any type of telecommunication network, including public and private, mobile and fixed telephone networks. Through Internet-telephony integration, some applications/services, normally implemented in CTI systems, can be offered in the public domain [AVT97], specifically within IN, as suggested in [Mine98] and [Webe98], including the use of CT APIs, for example JTAPI. This is useful for businesses which have single agents scattered in many locations, or for companies using a Virtual Private Network, e.g. for implementing the ACD application. However, the power of Internet-telephony integrated services lies in their potential accessibility by any end user with access to the Internet and the possibility of resource sharing across the whole Internet domain. An example that illustrates both these features is the Unified Messaging application, which can be offered publicly (to any user of the Internet), with the resources needed for the application on a server shared by numerous users.

Telephony and Internet functionality may be integrated at several different levels: at the level of management and configuration, at the service/application level and at the media level. In the area of management and configuration it is the telecommunication network that is managed and configured from the Internet, mostly in user service customisation and third-party service provision. An example of this can be found in [Burg98]. Developments have taken that direction owing to the specific natures of the Internet and the telecommunication network, whereby the Internet functionality is information and processing intensive, while the telecommunication networks' primary goal is to provide a high-quality, specific communication service[15]. Integration on the service/application level brings together service functionality of the telephone network and Internet application functionality. Integration at this level, which is examined in [Gull98], [Irvi98] and [Low98], could simply mean a request from one domain for call-control activities in the other or actual call-control exerted over one domain by the other. It could also mean synchronisation of some activities in the two domains in order to provide meaningful functionality, for example in the case of parallel communication of user information through the two networks. Finally, integration on the media level is in fact translation of information streams conveyed by the Internet into information streams suitable for transmission and delivery in the telecommunication network, in order to provide end to-end communication between users/devices on the Internet and users/devices on the telecommunication network. Systems of this kind are mentioned in [Bern98] and [Zorn98]. Most applications provided through Internet-telephony integration involve some combination of these three types of integration. The rest of this chapter concentrates primarily on the call-control level of Internet telephony integration, as does the whole of this work.

A point that must be made here, especially in the context of this work, which deals with Intelligent Networks, is that the most promising point for seamless and open integration of the telephone network with the Internet is within the "intelligence" contained in the IN [Low96].

[15] Management and configuration of Internet services by the telecommunication network, although possible, is quite impractical.

3.4.2 Internet –Telephony Integration Applications

This section contains the description of some applications offered through Internet-telephony integration, with notes on whether and how they relate to CT applications.

Click-to-Call-Back is an application whereby the user can click on a link on a company's WWW page to request to be called by a company agent. In fact, the communication with the web page is a bit more than a simple click of the mouse, since the user is expected to fill in and submit a form with at least his/her name and phone number but possibly other data. The user may request to be called straight away or at some other time. The WWW server, which must be Internet-telephony enabled, issues a request to the telephone network, which may be public or private, depending on the implementation and the requested call is placed. The telephone network side of the application must handle the search for a free agent, the alerting of the agent and the queuing of requests if necessary. This application uses Internet Telephony integration at the call control level.

Fax on Demand, which is traditionally a CT application initiated from a phone, in the Internet-telephone integration version allows the user to "click" on a link of a web page in order to have a fax sent to him/her. The click, as above, is not just a click: the user will have to fill in and submit a form, with his/her fax number and perhaps a choice of the actual document that he/she wants to receive from a particular WWW server on the Internet. An Internet Fax Server is approached with the request, so it fetches the fax document from the specified server and sends it over the telephone network. This application involves integration on the call control level and on the media level. It is interesting to note that the call control in this case could be first-party or third party, depending on whether the fax server is setting up the call directly or by sending a request to the switch in the telephone network.

Interactive Voice Response is a CT application, which requires special resources i.e. an IVR unit. The IVR unit's behaviour is controlled through a script. A telephone-banking IVR script may, for example, prescribe the playing of a message (prompting the user to enter a PIN), DTMF tone detection (recognising the entered PIN), a request for information from a database (with the PIN as the key), ASCII to voice conversion (of the retrieved database information) and the replay of the result of that conversion to the user. In the Internet-telephony version of this service, the IVR unit is in an Internet IVR server. IVR scripts are provided by different subscribers, at their respective WWW servers, in the form of html documents. The

html used to create these documents is a modified version, which includes tags for playing sound files and the like. This service makes use of Internet-telephony integration on the media level.

Similar to IVR is **Unified Messaging**, which allows all the messages for a user to be kept in one mail box, whether they be email, fax or voice mail messages. An Internet Unified Messaging Server is connected to the telephone network and has the capability to receive and process facsimile and voicemail messages. It receives email messages through the Internet. This service must include activities in the telephone network in order to redirect any facsimile calls and forwarded voice calls to the Unified Messaging service. Therefore, the telephone network would most likely be an Intelligent Network. This service involves media level integration, whereas the call-control, although a part of the service, takes part only in the telephone network, that is, the IN.

3.4.3 Implementation

Several different systems integrating the Internet and the telephone system have been implemented to date. Some of these systems have been designed "from scratch" for the purposes of Internet-telephony integration, some have been based on CTI and others have made use of IN.

An example of an Internet-telephony integrated system designed "from scratch" is the AT&T system [RFC2458]. It employs a logical entity called the CallBroker, which communicates with the telephony part of the system on behalf of the user on the Internet, to set up sessions and make requests for telephony resources. The CallBroker also performs user authentication, usage recording and other functions. Two interfaces are defined, the one between the IP client at the user's site and the CallBroker, and the one between the CallBroker and the telephone network. The protocol used between the IP client and the CallBroker is a text-based protocol which uses TCP for transport and allows full two-way communications.

The Nortel system, described in [RFC2458] relies on a protocol called the Simple Computer Telephony Protocol (SCTP), which is used for communications between an SCTP server and an SCTP client. SCTP is a third-party call-control protocol and the SCTP client uses it to request call-control operations from the SCTP server. The SCTP is usually connected to the telephony system through a CTI link. Apart from call-control, SCTP provides for abstract feature control (e.g. do-not-disturb) and monitoring of terminal, address and line status, which are typically parts

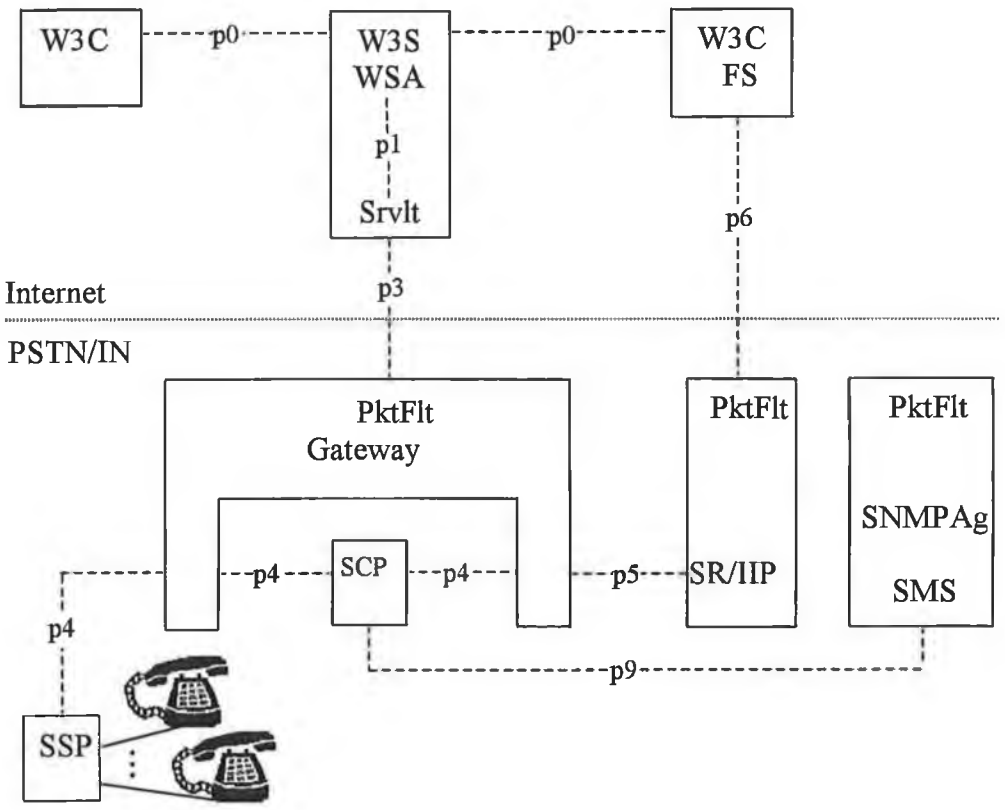
of CT functionality. The SCTP client can be embedded in a WWW server and in that case the user communicates with the system through http transactions. The client can otherwise be based in the user's PC, with it's own user interface and the possibility for two-way communication.

Two examples of IN-based Internet-telephony integration systems are the Siemens Web Call Centre and a system implemented by Lucent Technologies. Figures 3.3 and 3.4, respectively, show diagrams of those two systems [RFC2458].

3.4.3.1 The Siemens Web Call Centre

In the design of the Siemens system the aim was to hide from the SCF the fact that it was interacting with entities other than those belonging to the standard IN architecture. This was achieved by placing the Internet-IN gateway function on all the signalling paths between the SCF and the other functional entities, as shown in Figure 3.3, in order to intercept signalling messages and adjust them so as to "trick" the SCF into thinking that it was communicating with the SSF and SRF only. However, the SCF is actually also communicating with the WWW Server and through it, with the WWW Client and the user on the Internet.

For example, a Web Call Centre customer may request to be called back by a Web Call Centre agent at a certain time, by filling in a form and submitting it to the WWW server. The WWW server processes the form according to a CGI script and then sends the data collected in that way (e.g. customers telephone number and time at which to call him/her back) to the gateway. The gateway emulates a trigger message from the SSF and when the SCF, in accordance with the trigger, sends a request for a connection between the SSF and the SRF and a request for an announcement and data collection, those messages are "swallowed" by the gateway, which instead sends to the SCF, emulating a message from the SRF, the data already received from the WWW server. At this point, the SCF stores the request for a call-back and sends a message to the SSF instructing it to continue processing. This message also never reaches the SSF, since it is stopped in the gateway. Thus the functionality of the IN functional entities can be preserved almost in its entirety, while they are allowed to interact with the Internet. This method of integration works well where the IN set-up must not be usurped and where the number of services provided is not large. The gateway is service-specific and a greater number of services or more complicated services make for a complex and ultimately very impractical gateway.



W3C – World Wide Web Client
W3S – World Wide Web Server
WSA – W3S "back end" program interface (Interface towards CGI or Servlet)
Srvlt – Servlet "back end" program
FS – Finger Server
PktFlt – Packet Filter (Firewall)
INAP – Intelligent Network Application Part
INAPAd – INAP Adaptor
SNMPAg – Simple Network Management Protocol Agent
SR – Special Resource
IIP – "Internet" Intelligent Peripheral
SMS – Service Management System
SCP – Service Control Point
SSP – Service Switching Point

Protocols:
p0 – HyperText Transfer Protocol (HTTP)
p1 – HTTP Server <-> "back end" program internal protocol
p3 – PINT User Agent <-> PINT Gateway protocol
p4 – INAP
p5 – Proprietary (INAP-based) Gateway <-> IIP protocol
p6 – Finger protocol
p9 – SMS <-> SCP protocol

Figure 3.3 – The Siemens Web Call Centre

The other Internet end-user in this system, the first being the Web Call Centre customer, is the Web Call Centre agent. The Internet communications of the agent involve a reminder, when the time comes, of the fact that a customer has to be called back. The reminder is initiated from the system and for that reason cannot be implemented as an HTTP transaction, since HTTP transactions are always initiated by the client (the browser). It is implemented by using a Finger daemon process

running in the agent's computer, which receives Finger queries from an Internet Intelligent Peripheral (IIP), a functional entity specially introduced for the needs of IN Internet integration. These queries are triggered by Initiate Call Attempt requests from the SCF to the SSF, which are intercepted by the gateway and turned into appropriate requests to the IIP. On receiving the Finger query from the IIP, the daemon in the agent's computer sends the standard Finger response but also instructs the Web browser to start certain HTTP transactions (e.g. to download a reminder window, which may include information about the customer who is to be called back). Thus some communication in the direction towards the IP client is made possible.

3.4.3.2 The Lucent System

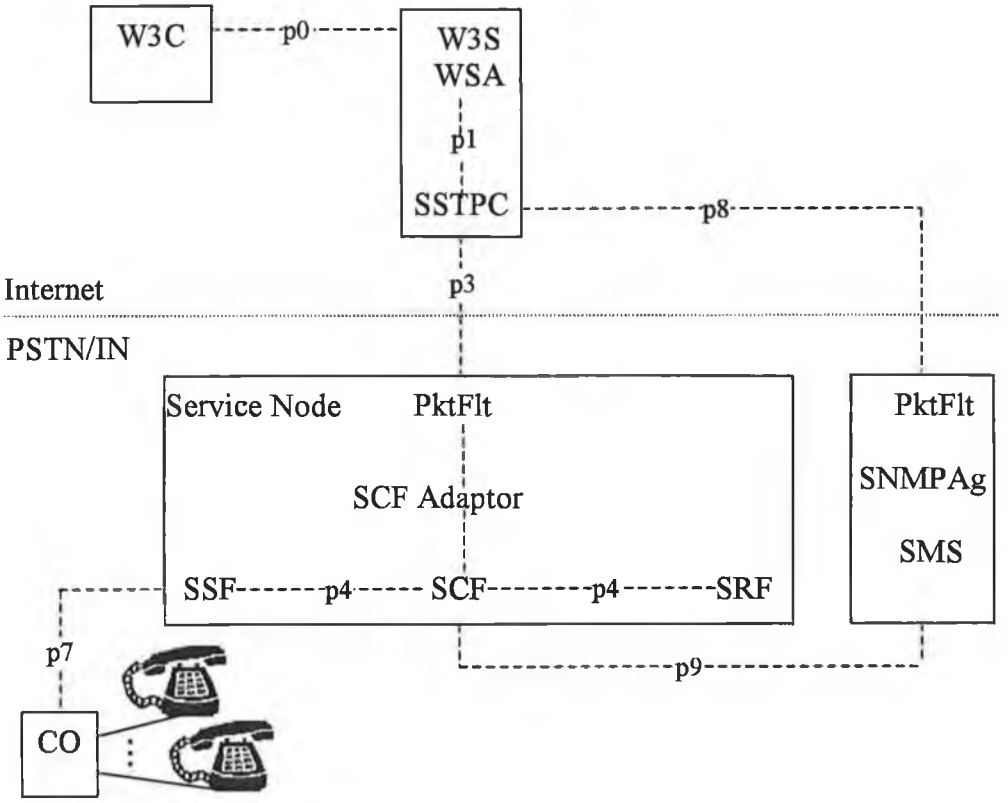
In contrast to the Siemens approach, here the SCF is fully aware that it is interacting with the Internet and its functionality has been adapted to that, extended IN environment. This system is built with a Service Node (SN), which encompasses SCF, SSF and SRF functionality, a Service Management System (SMS) and a Web server.

In the Click-to-Dial-Back service scenario, an end-user browsing the WWW page of a company may request to be called back by a representative by submitting his/her phone number and other information in an HTTP transaction with a Web server. The Web server sends a message with this request to the SN, including the information about the requesting user and the company (the Web content provider/IN subscriber). The SN then calls both sides and connects them, playing announcements where necessary (if one side is waiting to be connected).

The role of the SMS is to replicate the content provider/subscriber data, contained in the Web server, in the SN and to establish associations between those data and any subscriber-specific IN service data, contained in the SN (e.g. for time of day dependent routing). The subscriber-specific data have to be used in the SN-based part of services such as Click-to-Dial-Back.

The communication between the Web server and the SN uses the Service Support Transfer Protocol (SSTP), developed by Lucent. SSTP uses TCP for transport. The types of messages sent from the Web Server to the SN are Transaction Initiator, Data Message and End of Data Message. The last two types of messages are used in cases where the Web server provides the data to be sent through the telephone network, e.g. fax data for the Click-to-Fax service. The communication between the

Web server and the SMS uses the standard Simple Network Management Protocol (SNMP).



W3C – World Wide Web Client
W3S – World Wide Web Server
WSA – W3S "back end" program interface (Interface towards CGI or Servlet)
SSTPC – Service Support Transport Protocol Client
PktFlt – Packet Filter (Firewall)
SNMPAg – Simple Network Management Protocol Agent
SMS – Service Management System
SCF – Service Control Function
SSF – Service Switching Function
SRF – Special Resource Function
CO – Central Office/ Public Telephone Exchange

Protocols:
p0 – HyperText Transfer Protocol (HTTP)
p1 – HTTP Server <-> "back end" program internal protocol
p3 – PINT User Agent <-> PINT Gateway protocol
p4 – Intelligent Network Application Part
p7 – Digital Subscriber Signaling 1
p8 – Simple Network Management Protocol
p9 – SMS <-> SCP protocol

Figure 3.4 – The Lucent PSTN-Internet Integrated System

3.4.4 Standardisation

This section looks at activities in the direction of standardising Internet-telephony integration at the call-control level. Such activities are taking place in both the Internet and the IN communities. On the IN side, it is planned that ITU-T Capability Set 4, which should be completely specified by the end of 1999, will

support the possibility of service requests from the Internet. ETSI are working on Next-generation IN systems and extensions to Core-INAP which will provide input to the ITU-T work in the area of Internet integration, among others [Conr99].

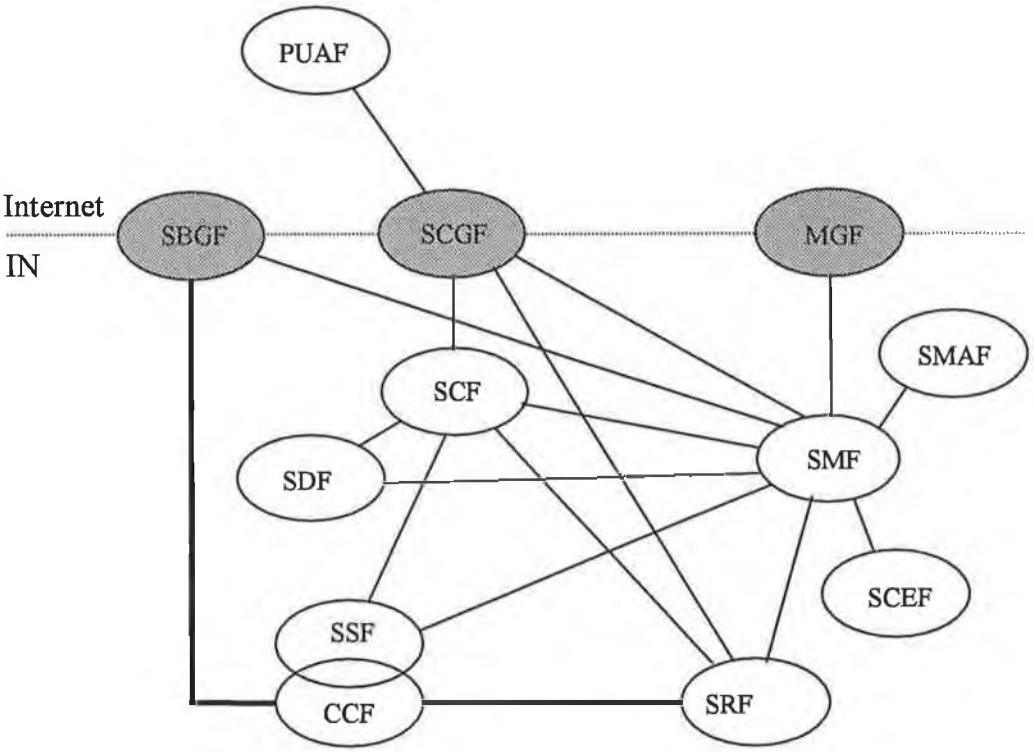
On the Internet side, the Internet Engineering Task Force (IETF) started the PSTN and Internet Internetworking (PINT) Working Group, in 1997 [PINT99]. Although its name implies it, PINT is not limited to Internet integration with PSTN but aspires to apply to any type of telephone network e.g. PSTN, mobile and private telephone networks.

The aim of the PINT WG is to study the protocols in the Internet and architecture needed to enable the user to request telephone network services through the Internet. It aims to define a protocol, operating over TCP, to accommodate the communication between the entity serving the Internet part of such services and the Service Control Function (SCF) or some equivalent entity in the telephone network. The protocol is not to support any type of call control but to carry requests from the Internet to the telephone network. Another aim is to define a Management Information Base (MIB) extension of managed objects to be used by the standard Simple Network Management Protocol (SNMP) when managing the system providing such services. Further, the PINT WG is to investigate security issues in such systems and possible solutions.

Initially, four specific services will be taken into account in this context. They are Click-to-Dial-Back, Click-to-Fax, Click-to-Fax-Back and (Internet-Initiated) Voice-Access-to-Content. The PINT WG intends to extend the architecture and the protocols to support a wider range of services in the future.

At the moment the Session Initiation Protocol (SIP) is being considered within PINT as a possible solution. SIP has been defined by the Multiparty Multimedia Session Control (Mmusic) Working Group of the IETF and is used to establish the association between participants within an Internet conference call. It is being extended in order to support calls in the telephone network.

Figure 3.5 shows the DFP architecture for the IN integrated with the Internet, currently in draft, including a SIP link between the IN and the Internet [Lu98].



CCF - Call Control Function
SCEF - Service Creation Environment Function
SCF - Service Control Function
SDF - Service Data Function
SMAF - Service Management Access Function
SMF - Service Management Function
SRF - Specialised Resource Function
SSF - Service Switching Function
C/BGF – Call/Bearer control Gateway Function
SCGF – Service Control Gateway Function
MGF – Management Gateway Function
PUAF – PINT User Agent Function

Bearer Connection Control
IN Service Control
Management Relationship

NOTE: The Functional Entities depicted in gray are new to IN.

Figure 3.5 – ITU-T Draft for Internet-IN Integrated System

3.5 Conclusion

This chapter has described the existing systems that integrate telephone network and computer network functionality. First of all, there is Computer Telephony Integration, which, in the case of systems with third-party call control and shared resources, is the private/local area network analogue of an IN/Internet integrated network. Computer Telephony systems are therefore a good source of service ideas for an IN/Internet integrated system, such as the one designed in this project.

Further, possible applications for a system integrating the public telephone network and the Internet are described, as well as implemented systems, which exist in various proprietary forms. The general structure of the systems is similar, with a central link between the SCF, or its equivalent, to the Internet.

Although there have been activities towards standardisation of the Internet-specific aspects of Internet – telephone network integration in the Internet community, there have not yet been any definite telecommunications-oriented activities in that direction. This project addresses IN/Internet integration primarily and in detail from the point of view of IN and takes into consideration the standard incremental manner of introducing new functionality into IN, identifying potential elements of the integrated system that could be standardised. However, the general structure of the system, need not be different from that of the existing systems.

4 The Internet Integrated Intelligent Network

4.1 Introduction

As mentioned in the introduction to the thesis, the aim of this project is to explore the possibility of logically incorporating the Internet and its resources, primarily as a means of real time communication and service control, into the Intelligent Network. Specifically, a system named the Internet Integrated Intelligent Network (IIIN) has been designed and a prototype of it built to that end.

The basic idea presented here envisages a single[16] physical link between a telephone network, namely an IN, and the Internet, the two most widely established publicly accessible networks. This link, albeit in a roundabout way, automatically provides a physical connection between a telephone and a PC lying side by side at each end user's desk. Such a physical connection opens the possibility for the synchronisation, combination and mutual influence of the behaviour of a telephone and a personal computer associated by location and thus for the provision of some new functionality without requirements for new types of terminal equipment. Therefore, the basic idea is to build a system that will offer the user, who has access to the two pieces of terminal equipment, functionality not offered by that system's two constituent parts, IN and the Internet, as they are now.

While the use only of existing terminal equipment and the provision of new functionality through the logical combination of the interfaces that that equipment offers to the user has been the aim of the project, the system has also been designed so as not to impose any requirements on the user for purchases of expensive software. Most of the functionality is designed to be implemented as a part of the network, while the software required by the end-user is either that already widely existent among Internet users (e.g. Web browsers) or has been designed to be light-weight and free of charge.

[16] The word 'single' here should not be taken literally. We are assuming that the IN has one SCP but any upward scaling of the described system would, of course, involve several such links.

Two crucial characteristics of this first-instance IIIN, one functional and one implementational, are evident. First, in integrating the Internet and the IN, it ultimately considers both networks as communication vehicles for the end user, unlike systems where, for example, the Internet is used as a distributed resource-pool for the IN [Irvi98]. Second, the integration is designed to be non-intrusive and the interface between the IN and the Internet is minimal. (A case of intrusive integration, with service logic of the IN being replaced by distributed functionality in the Internet, is described in [Low96] and the signalling gateway defined in [OMG98] is an example of introduction of Internet technology into the IN, which allows for seamless but entirely intrusive integration.) Integration of a type similar to what is dealt with in this project is described in [Webe98] and [Bern98].

The type of Internet-telephone integration studied here is integration at the service/application level (see §3.4.1), whereby already existing or specifically designed IN services and Internet applications co-operate to provide new functionality. The forms that this co-operation takes are call-control, requests for service from one network to the other and synchronisation of information flows in different networks. This project has for its task to recognise the functionality that can be provided in this way and to suggest some rules for cross-network co-operation, in the form of architecture and protocols.

It should be noted that in this project the integration of the Intelligent Network and the Internet is studied primarily from the point of view of the Intelligent Network. The system has been described in terms of existing IN architecture and the accent of the work has been on designing a method for integrating the two networks, which would minimally disturb the architecture of IN as it is now. Apart from the special interest the author has in Intelligent Networks there is another reason for this approach and that is the (yet) closed nature of telecommunication networks in contrast to the openness of the Internet. It is logical to start with designing an integrated system on the more limiting side (in this case the IN) and then to take advantage of the less restricting side (in this case the Internet) to "pick up the loose ends".

The Internet Integrated Intelligent Network (IIIN), therefore, denotes an Intelligent Network, extended to include the link to the Internet and the entities on both the IN and the Internet side which provide IIIN-specific functionality. Therein is included functionality which is really Internet application functionality, albeit pertaining to IN. However, for the needs of this project, the Internet is logically

"hijacked" on behalf of the IN and viewed as a freely available resource to be used by the IN, just as is, for example, the computer used to implement the SCF together with its operating system. Therefore, we will be speaking of IIIN services (extended IN services) rather than "Internet applications co-operating with IN services". The description of the IIIN starts with a look at how the work done on this project relates to the IN Capability Sets.

4.2 The Internet Integration Capability Set Increment 1

4.2.1 IICSI1 in the Context of IN Capability Sets

The Internet Integrated Intelligent Network (IIIN) is a system that integrates the Intelligent Network (IN) and the Internet. The IIIN is defined as the IN at some unspecified stage of development (e.g. CS2), extended by one or more specifically defined capability set increments, called the Internet Integration Capability Set Increments (IICSIs), which each consist of some set of exclusively Internet-integration-specific capabilities. The overall capabilities of the resulting IIIN vary depending on the actual CS and the set of IICSIs that it has been built upon. The IICSIs have been envisioned as potential contributions to some concrete future capability set specifications.

In this project, the IIIN has been described as CS1 IN extended by one IICSI, the IICSI1. CS1 has been used as a base for this description of the IIIN because of its familiarity and relative simplicity. However, the set of capabilities offered by an IIIN based exclusively on CS1 would be very limited. The need of the described IIIN for CS-2 capabilities is noted in the text, wherever appropriate and the required capabilities are described.

4.2.2 IICS1 in the Context of Internet-IN Integration Types

4.2.2.1 Internet -IN Integration Types

This section briefly describes the types of IN-Internet integration that have been identified. As mentioned in Chapter 3, the integration between the Internet and a telephone network can take place at three different levels of activity in the networks. Figure 4.1 illustrates these three types of integration, applied to IN. In the case of integration at the management level, it is the Service Management Function that is in

some way co-operating with the Internet. At the service/application level, the Service Control Function is exchanging service control information with the Internet. Finally, at the media level, it is the voice channel information that is translated into Internet Protocol traffic. The SRF, being an entity handling voice-channel information, has been included as a candidate for information exchange with the Internet at the media level. Each of these types of integration requires some kind of a gateway between the appropriate functional entity in the IN and the Internet to support the exchange of information at the corresponding level.

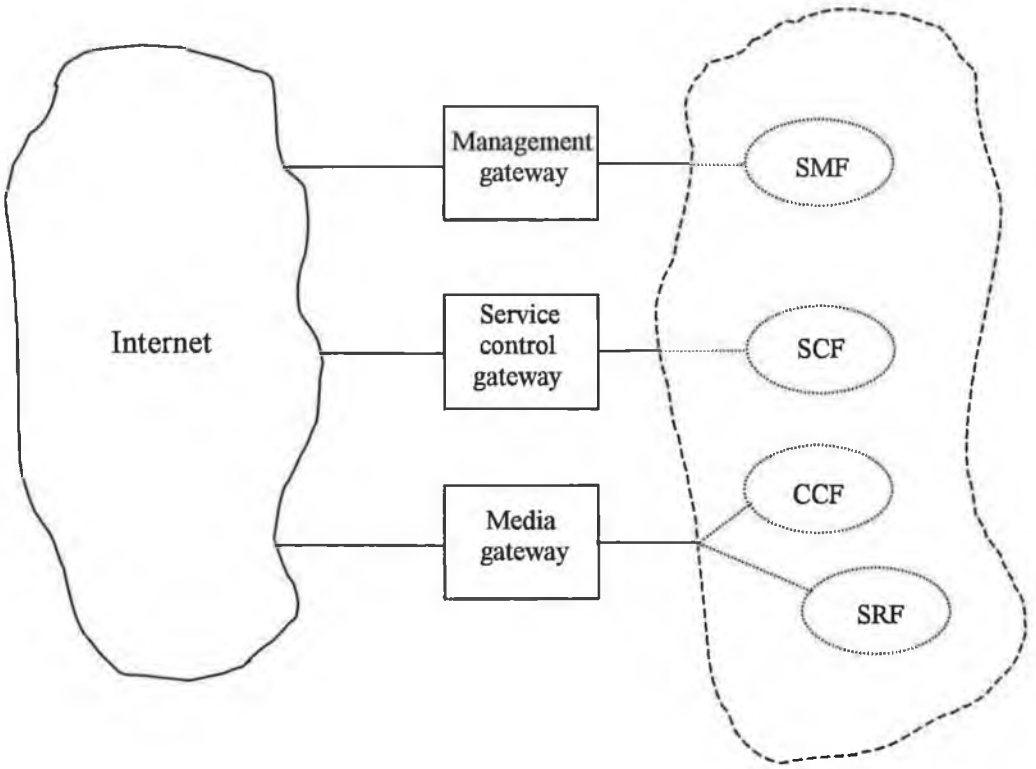
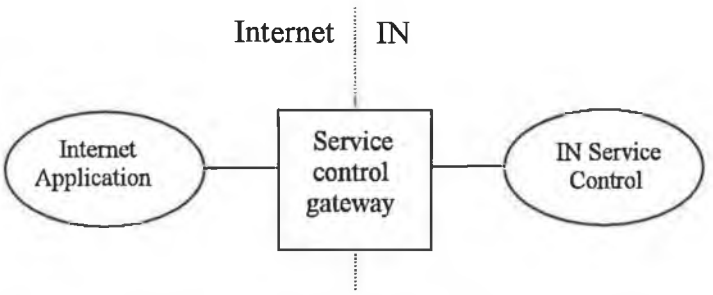


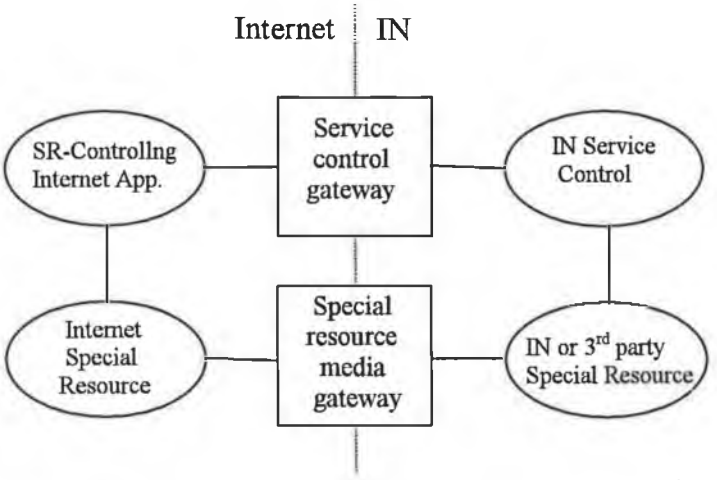
Figure 4.1 – Gateway Types for IN-Internet Integration

The gateway at the media level includes the actual translation of the media information streams, but also has a non-media component – the translation of routing information and other signalling (e.g the translation of ISDN call setup messages). This functionality, when part of the telephone network alone, is not a concern for IN. On the other hand, the gateway itself introduces translation functionality that might be implemented in the form of IN services, and therefore might use the service control gateway in the figure. This project, however, does not deal with that case. In the specific case of a Special Resource, which, even in its connection control aspect, is a concern of IN, the functionality that a media gateway connected to it might make possible is examined and related requirements from the point of view of service

control identified. As to the other two types of integration, the management gateway is not studied as part of the project at all, while the study of the service control gateway is its central objective. Figure 4.2 illustrates the two types of integration that are supported by the IICSI1.



a) Integration at the service control level



b) Special resource media-level integration with assisting service control-level integration

Figure 4.2 – IN-Internet Integration Types for IICSI1

4.2.2.2 Integration Types for IICSI1

To summarise the above, the group of capabilities contained in IICSI1, and studied in this project, deal mostly with the functionality made possible through a service control gateway between the IN and the Internet. It also deals with service control functionality associated with the media gateway between the Internet and the special resource in the IN.

These two types of gateway between the IN and the Internet allow for services to employ different types of co-operation during their execution, through a minimal interface in the case of IIIN. This section describes the subset of such co-operation

types (for each the Service Control Gateway and the Special Resource Gateway) that defines the IICSI1. These forms of co-operation are categorised – and illustrated in Figure 4.3 – in terms of flows of information through the service control-level gateway.

4.2.2.2.1 Service Control Gateway in a Standalone Role

First of all, a connection to the Internet may be used as a means of activating services in the IN. In this way the Internet may be used to replace some IN functionality, especially that of user interaction with the special resource, so that services that could otherwise be provided by the IN alone are made more user friendly through the use of the Internet. An IN service feature especially suitable for such adaptation is the Customer Profile Management (CPM) feature, where all the collection of relevant data could be done in the Internet, then a service feature activated in the IN to accept and process the data. Apart from these services that can be provided by pure IN but are enhanced with the Internet, there are numerous services with "added value" that are based on service activation in the IN from some entity in the Internet. Those are, for example, Click-to-Dial-Back. Here a user requests, through the Internet (using a CGI form or some other mechanism) to be called back on the telephone, by the owner of the Web page, e.g. an insurance company, represented by an agent. The Web server processes the request and conveys it – through the service control gateway-to the IN, which places the appropriate call in the telephone network and connects the agent to the user. The flow of information between the Internet and the IN for this group of services is shown in Figure 4.3a.

Another type of co-operation between the service control in the IN and the Internet is that carried out to co-ordinate real-time events in the two networks. An example service that should make use of such co-ordination is Parallel Routing, described later in this chapter. Figure 4.3b shows the flows of information that move through the service control gateway when this type of co-operation is taking place: an entity in the Internet requests some dynamic information from the IN and receives a reply. Flow in the other direction, with dynamic information being sought from the Internet on the part of the IN, is also possible. However, such services have not been considered as part of IICSI1 (see §4.2.2.2.3).

Next, the SCF may invoke some activity in the Internet, as part of a service. For example, an Automatic Call Distribution (ACD) application, implemented in IN, may send requests to some entity in the Internet for pop-ups of information about the

calling customers to be sent to the agents' screens. The appropriate flows of information are shown in Figure 4.3c.

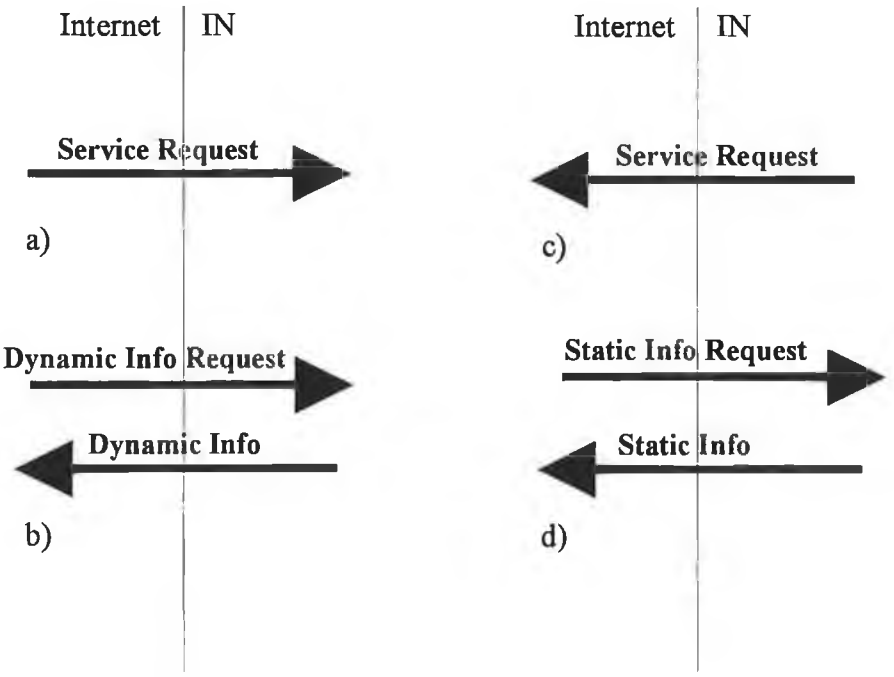


Figure 4.3 – Types of Internet-IN Co-Operation

Finally, the co-operation may take the form of a request for static information pertaining to an IN user, from the IN, by some entity in the Internet. For example, a service called Email-by-Number could allow a user to send an email to someone without knowing their email address, but by specifying the recipient's telephone number as the address. The database matching telephone numbers with email addresses could be based in the IN, so the provider of the Email-by-Number service in the Internet must query the SCF. The information exchanges between the Internet and the IN for this case are shown in Figure 4.3d. The same type of exchange, but in the opposite direction, is another possibility – which, however is not an IICSI1 capability (see §4.2.2.2.3).

4.2.2.2.2 SR Media Gateway Supported by Service Control Gateway

Some clarification on what is here considered to be a special resource media gateway: any media gateway that does not provide real-time end-to-end connectivity between a terminal device on the IN and an IP host is classified here as a special resource media gateway. Opposed to a special resource media gateway is a call

control media gateway, which does have the role to provide such real-time, end-to-end connectivity.

Central to the special resource media gateway is a group of Computer Telephony (CT) applications, which can be offered in the public domain with the use of the Internet as the public equivalent of the LAN and the IN as the public equivalent of the PBX. Examples of such services are Fax-on-Demand, Interactive Voice Response and Unified Messaging, all described in Chapter 3 (§3.4.2). These applications involve routing of incoming calls to a special resource, which has access to various resources on the LAN. The routing, performed in CT either by the PBX itself or by the CT application, would be performed, in the Internet – IN arrangement, by the IN. The special resource would have access to the Internet and its resources through the special resource media gateway. The IN may or may not exercise service control in the implementations of such services. However, it is the implementations of such services that do include communication through the service control gateway that are considered here.

The special resource media gateway does not apply only to special resources in the IN sense of the word. Such services could be offered by an independent service provider, who would also provide the special resource hardware, appearing to the IN as just another terminal in the telephone network, but connected at the "back end" to the Internet. This type of special resource will henceforth be referred to as a third party-provided special resource or a third-party special resource.

In the case of a third party-provided special resource, routing of the call would take the form of an ordinary phone call or, at the most, a call involving a number translation. Nevertheless, the IN would have to have knowledge of the special function of this terminal device, in order to retain and give it information about the call that it may need in order to provide the service. The flows of information expected through the service control gateway during activity related to a third party special resource gateway are the same as in figure 4.3b, the dynamic information being requested by the third party-provided special resource's back-end program in the Internet.

A special resource in the IN is controlled within the IN by the IN service control function. However, the entity on the Internet side, communicating with the IN special resource through the IN special resource media gateway, must be controlled as well, and that is done by the service control part of the IN through the service control gateway. The type of co-operation between the Internet and the IN during an

instance of a service involving an IN special resource media gateway is as illustrated in 4.3c.

An example of a service using the special resource media gateway is Unified Messaging. Unified Messaging allows a user to receive all types of messages, such as e-mail, fax and voice mail, in the same mailbox. If another user dials the user's facsimile number (which, in the case of this service, does not correspond to any physical location), the call is routed to the fax – receiving special resource. Once it has received the fax, the special resource turns the received information into data transferable over the Internet and sends it to the appropriate mailbox on the Internet. The address of the mailbox will have been received from the IN service control functionality, which is able to determine the address from the originally dialled number. In the case of a third party-provided special resource, the device contains a controlling program which requests the Internet destination address from the IN once it receives the data from the telephone network. In the case of an IN special resource, the IN first instructs the special resource to receive the data and feed it into the gateway for format conversion. Then it instructs the special resource's Internet counterpart, through the service control gateway, to send the received and converted data to an address in the Internet. Voicemail messages, for the same service, are received in a similar manner, except that the nature of the used SRF is to receive voicemails rather that faxes. Emails, of course, are received in the usual way, without the involvement of IN.

It should be noted that the special resource gateway functionality could, alternatively, be realised exclusively in the switching domain, where the special resource gateways would act rather like "Internet PBXs" handling a certain number of telephone-network address - Internet address mappings. In that case the IN could be employed to administer special premium-rate or freephone-number-like billing arrangements. Such service solutions are not subject matter for this work.

4.2.2.2.3 Restrictions on the Use of IICSI1 Integration Types

An important point to be made here is that in all the cases described above where the Internet initiates an information exchange (a, b and d), a new instance of service logic is started in the IN. No influence from the Internet, synchronous or asynchronous, on the IN service logic is allowed after the service logic is started. This restriction has been introduced to narrow the IICSI1, as the first step in Internet-IN integration, in a way that reduces performance and security concerns. This is, also,

the reason why the cases b and d do not have counterparts in the opposite direction for IICSI1. This restriction results in a non-intrusive nature, mentioned earlier, of the IIIN integration solution.

This restriction also means that calls in the IN cannot be manipulated from the Internet. Since earlier in this chapter the IIIN has been pronounced to be a public system analogous to Computer Telephony in the world of private networks, it should be noted that the analogy is somewhat reduced because of it. The true call control features present as an important part in CTI are excluded from IICSI1.

4.2.2.3 Non – IICSI1 Integration Types

For clarity, the non-IICSI1 Internet-IN integration types are described here briefly, although they are not the subject of this project.

4.2.2.3.1 Call Control Media Gateway Supported by Service Control Gateway

The call control media gateway allows the flow of a media information stream, primarily speech, in real time, between an end user in the IN and an end user in the Internet. This will, in most cases, mean the translation of a PCM encoded voice signal into IP-carried information, like Voice over Internet Protocol (VoIP), and cross-network call routing. While the media information stream translation is not a concern for IN, some of the routing functions could be implemented in IN, not unlike their equivalents in the mobile world, which requires relatively complex service processing for simple end – to – end connectivity. This would involve service control communication between the SCF and an equivalent entity in the Internet.

4.2.2.3.2 Service Management Gateway

The IICSI does not cover any aspect of management or service creation in the IN or, indeed, any use that their functionality might have for Internet resources. However, management in the Intelligent Network could incorporate Internet resources in several different ways. For example, the Service Creation Environment Function (SCEF) could be located anywhere on the Internet and created services could be sent to the Service Management Function (SMF) for deployment. Also, the Service Management Access Function (SMAF) could be located anywhere, while communicating through the Internet with the SMF. Finally, the SMF could actively co-operate with management entities on the Internet to perform its functions, especially in the case of IIIN services.

4.2.3 Integration Modelling of the IIN for IICSI1

It is tempting to try to model the IIN as a symmetrical system, presented in Figure 4.4. However, no deep analysis of the system is required in order to conclude that such a model would be entirely inappropriate. The Intelligent Network is, in essence, a specific solution to the problem of introducing "intelligence" i.e. information processing power, into a network of mostly "dumb" terminals. Whether centralised or distributed, the intelligence governing the behaviour of such a network is different in nature, and therefore separate, from the network. Its possibilities for manipulating the network are limited by the capabilities of the terminals (telephones) and their standard behaviour (end-to-end connectivity). On the other hand, the Internet is a non-centralised system consisting of independently functioning computers, each powerful in their own right and capable of performing an enormous variety of functions. The computers communicate among them in order to achieve additional functionality through sharing both information and processing capabilities. There is, obviously, no need for service logic to control the activities in the Internet, since any amount of "service logic" can be incorporated into those activities themselves.

For these reasons, it is more useful to view the Internet capabilities as extensions to the service logic in the IN, as shown in Figure 4.5. IN does allow services to be invoked from the service logic, rather than from the Basic Call Process and this makes it possible for the events in the Internet i.e. users at terminals attached to the Internet, to become initiators of service logic. This is exactly what happens in the case of the Internet providing user interaction that leads to service initiation, as described in §4.2.3. In the case of static or dynamic information exchange between the IN and the Internet, all the communication takes place within the service control space of the IIN.

The model of the IIN for the case of IN-Internet integration through a third party-provided special resource gateway is as presented in Figure 4.6. The special resource is connected to the IN as any other telephone network terminal device. The recognition of a call to such a device is implemented as an ordinary IN service, which stores information needed by the special resource. Such information may be requested by the service control entity in the Internet manipulating the special

resource on the Internet side of the SR gateway and accordingly supplied by the service control entity in the IN, both through the service control gateway.

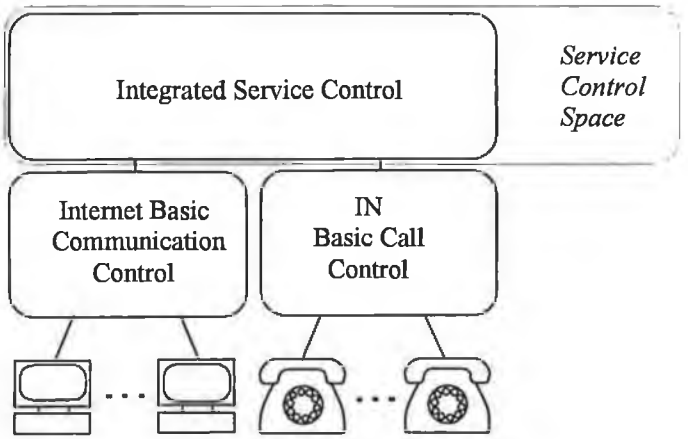


Figure 4.4 – IIIN – The Symmetrical but Inappropriate Integration Model

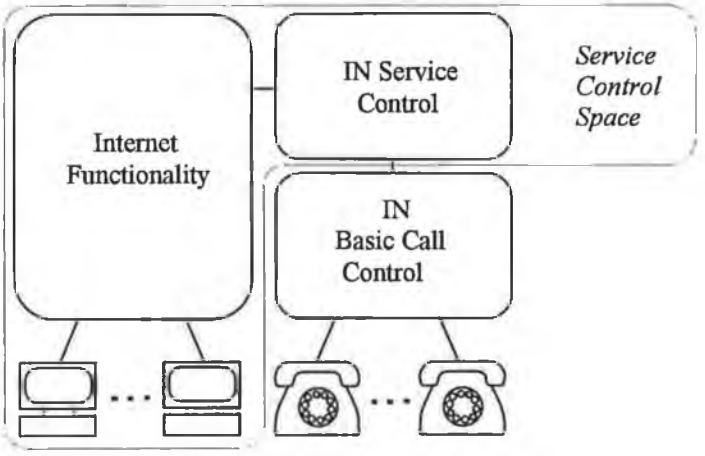


Figure 4.5 – IIIN Integration Model for Services Using the Service Control Gateway in a Standalone Role

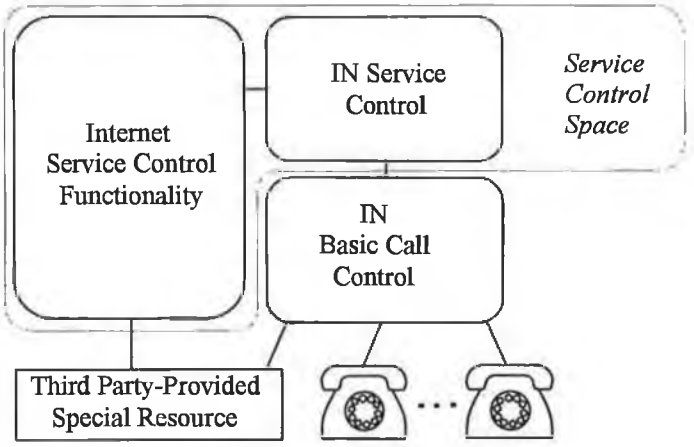


Figure 4.6 – IIIN Integration Model for Services Using the Third Party-Provided Special Resource Gateway

Finally, Figure 4.7 shows the IIIN integration model for integration through an IN special resource gateway. Here the Internet extends, in an abstract way, the IN special resource. The extended special resource is controlled by the service control entity through an extended set of operations.

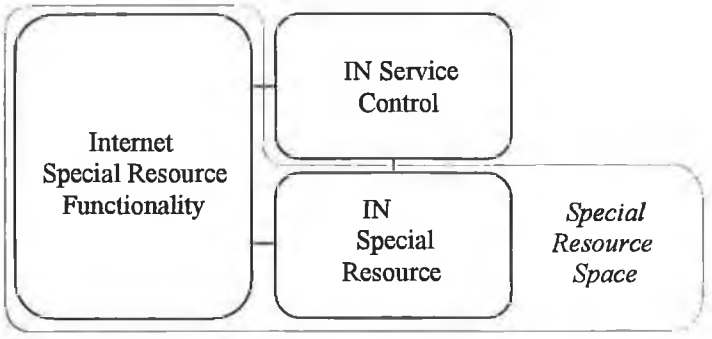


Figure 4.7 – IIIN Integration Model for Services Using the IN Special Resource Gateway

4.3 IICSI1 and the Intelligent Network Conceptual Model

4.3.1 Introduction

Since IICSI1 represents an extension to IN just as any other CS increment, it is natural to place the so extended IN, that is the IIIN, in the context of the IN Conceptual Model (INCM) and to describe it within each plane of the model.

The Internet part of the IIIN and the IN part of the IIIN are analysed separately, except on the Service Plane, since it is natural to view the IIIN services as wholes offered by that system. As for the other planes, the Internet and IN parts of the system have been separated for two reasons: the difference in nature between the Internet and the IN, which dictates different methods of description; and a need to clearly identify the elements that are to change in the IN itself, without simultaneously looking at, essentially independent, structures and functionality belonging to the Internet.

As will be seen in this chapter, one of the important changes that IICSI1 IIIN requires to be made to the IN is the ability to communicate with the service control gateway and optionally a special resource gateway towards the Internet. It also requires the enlargement of IN capabilities by a set of Service -Independent Building Blocks (SIBs), that is, the IICSI1 SIBs. Finally, it needs the addition of some new

messages on the relation Service Control Function (SCF) – Special Resource Function (SRF) and the addition of media-translation functions in the SRF, which are described but not implemented as part of the prototype.

The boundary between the Internet and IN part of the IIN lies in the two gateways, which are included in the description of each of the two parts.

4.3.2 Service Plane

This section gives a description of the services, which are to be offered by the most basic IIN, built by extending the IN CS1 with IICSI1. As mentioned before, there could be some exceptions to relying exclusively on CS1 capabilities, but those will be noted where they appear.

All the services described here are based on service control level integration and special resource media-level integration between the IN and the Internet, since those are the types of integration included in IICSI1. The services are grouped as follows: services using service control gateway standalone functionality, services using special resource gateway functionality (with supporting service control gateway functionality) and services using both of those.

The services listed below have been chosen as logical service units, which can be offered independently to the user. However, some of these services could be combined to form other, more complex services. For example, the Click-to-Dial and Automatic Call Distribution could be offered as members of a complex Call Centre service. On the other hand, some of the described services could be decomposed into service features, not meaningful as stand-alone entities, some of them not specific to the IIN. This work does not delve into the details of such decomposition and combination - and therefore will not provide any service to service feature mappings - but aims to define a set of capabilities, which manifest themselves in the services explained below. The only service feature (as opposed to services) described is Customer Profile Management and the reason it is considered separately is that it does not naturally represent a service itself but can be a part of many services, the same as Customer Profile Management in the standard IN.

4.3.2.1 Services Using the Service Control Gateway Standalone Functionality

Click-to-Dial is a service, which allows a user to request, through the Internet, that a telephone call be placed in his/her name to another user on the telephone network. The telephone number of the user to be called is contained in the

request, whatever form it might take (it could really be a click of a mouse on a WWW page link, or a form that is filled in and submitted by the calling user). The user that initiated the service is alerted first and connected to an announcement after answering, until the other user is called. The alerting can be either normal ringing or a message on the screen, coming from the Internet, prompting the user to pick up the receiver.

In a more elaborate version of this service, a user may request the set-up of a conference call from the Internet. With regard to the more complex data needed to request conference calls, the use of the Internet is a way of improving that service.

Click-to-Dial-Back is similar to the Click-to-Dial service. A request for a call to be placed in the telephone network is sent to the IN from the Internet. However, here it is the other party, usually an agent in a company that is to ring the user that requested the service at a certain time, also specified by that user. The agent is rung first and given an announcement after answering to wait until the user that requested the call is rung. In the case that a form with some additional information has been filled in by the user at the time of requesting the service, invoking a pop-up on the computer screen of the agent with the requestor's data, could be added as a feature to this service.

The **Group Announcements** service allows a user to make an announcement to a group of people, e.g. to tell all the members of a choir that a rehearsal has been cancelled. The choir conductor, who may be requesting the service, submits the list of numbers to be called through the Internet (most probably through the WWW). Then within, for example, 5 minutes, he makes a call to a certain number and gets connected to a special resource, which prompts him to say the announcement to be made to the choir members, and the announcement is recorded. After that, the IN proceeds with running a predictive dialling program, which calls all the numbers on the submitted list and plays the announcement. In another scenario, the user requesting the service creates an SRF script and submits it, together with the list. For the announcement, the script could either contain text to be converted to speech by the special resource or a sound file to be played by the special resource. The script file could also prescribe user interaction, such as, in our example, a prompt to the listener to state the preferred day for a substitute rehearsal, where part of the announcement could be: "press 1 for Tuesday, 2 for Wednesday, 3 for Thursday and 4 if you have no preference." After ringing all the numbers, or after a certain time, if

not all the people at those numbers have been reached, the result of the user interaction is emailed to the user who initiated the announcement.

A good idea is that the ringing pattern in the case of announcements be different from that used for ordinary calls, in order to warn the person answering that he/she is going to hear a machine speaking, rather than talk to a person.

Tele-voting is an extended version of the ordinary IN Tele-voting service. Tele-voting calls made from the telephone network are treated normally, however, if a user is specially subscribed to IIIN services, he/she receives a pop-up on the computer screen with information on the choices to be voted for. The user can then place his/her vote either through the phone or through the Internet. Another scenario is that the user actively accesses a WWW page from which he/she votes. All the votes, from the Internet and from the telephone network are added up as part of the same voting process.

Automatic Call Distribution is the same as that application in CT, except that it is offered in the public network. This version of the application is useful for businesses that have agents working from home, receiving calls through the public network. The service distributes calls coming in for the business to the agents that are free, on a round-robin basis. Through the Internet, a pop-up is sent to the agent's computer screen, with data about the calling customer.

Emergency Notification is similar to the Click-to-Dial service in that a request for a call is sent to the telephone network from the Internet. However, in this case the requesting party is not a person but a piece of sensor equipment connected to the Internet. A call is made to an appropriate number with an alarm announcement. For example, a smoke detector could be designed to send a request over the Internet for a call to the local police station, if a fire is detected. The announcement contains information on where the emergency has occurred and of what nature it is.

Alarm Call is like an ordinary alarm call to the user's telephone, except that it is set from the Internet rather than from the telephone. The user sets the time and number at which he/she wishes to be called and later, at the specified time, receives a call with an announcement.

Parallel Routing allows a user on a telephone call to transparently send data to the user at the other end of the call, from one computer to the other, through the Internet. An example of using this service is in an estate agent business. The agent is talking to a customer about a house. She clicks on a photograph of the house,

displayed on her own screen and it is automatically popped-up on the customer's screen, if he has a computer on his desk.

Email-by-Number is a by-product service of the IIIN. The IIIN needs to have a database with IN- and Internet-relevant data for users, and the email address could be included there. This makes it possible for people to send emails to users in this database by using the phone number rather than the email address as the destination address.

Calling Line Identification (CLI) is the same as CLI in "pure" telephony, except that the calling number is displayed as a pop-up on the called user's screen rather than on the telephone apparatus itself. This is a useful small service for users with phones without a display.

Customer Profile Management is a service feature. It allows the user to add and change his/her own user-specific data, which determines the behaviour of a service in the instances pertaining to that user. The changes are made from the Internet, most likely using a Web page form, which is filled in and submitted, then sent by the Web server to the IN for processing. This service feature can be part of many different services.

4.3.2.2 Services Using the SR Gateway Functionality

The services listed here either use a third party special resource gateway or an IN special resource gateway.

Unified Messaging has already been described in detail in §4.2.2.2.2.

Fax-on-Demand allows a subscriber to publish fax documents on the WWW and a user to retrieve those documents through the telephone network. The user is lead through a user interaction script whereby she enters the fax number of the machine at which she wishes to receive the fax and selects the document to be faxed, all by entering DTMF digits.

Interactive Voice Response (IVR) allows a subscriber to publish IVR scripts on the Internet. When a user dials the number corresponding to a particular document, the script is played out to the user. A special use of this service is to allow blind people access to information on the Internet and WWW browsing. HTML would have to be modified slightly to accommodate Web pages used with this application.

4.3.2.3 Services Using the SR Gateway Functionality and Service Control Gateway Standalone Functionality

In the case that the special resource is provided by a third party there may not be any need for the use of IN for some of these services (for example, it could be a special fax machine that receives requests and documents from the Internet and automatically faxes them to a specified number). However, if it is a special resource in the IN that is used, these services are relevant in the context of this project.

Request to Fax allows a user to request, through the Internet, that a fax be sent, specifying the document to be faxed, based in the Internet, and the destination number of the receiving fax-machine, based, of course, in the telephone network.

Request to Hear Content works in the same way as Request to Fax, except that the destination is not a fax machine but a telephone, the number for which is specified in the request. Also, the document that is being sent from the Internet is not converted into a fax document before sending, but into speech, either generated from a text file or directly from some type of sound file.

4.3.3 Global Functional Plane – IN Part

On the Global Functional Plane (GFP) the capabilities contained in the IN part of the Internet Integrated Intelligent Network (IIIN) are viewed in a service-independent manner. This section contains the description of the Global Functional Plane Model and of the Service Independent Building Blocks (SIBs) used in IICSI1. First the IIIN-specific SIBs needed to build IICSI1 services are identified and described. Then note is taken of the service-independent capabilities of the standard IN Capability Sets that are also essential to the construction of those services.

4.3.3.1 The IICSI1 Global Functional Plane Model

This model, shown in Figure 4.8, is to a certain extent the same as the model for IN (Figure 2.2 – GFP Part). However, it includes two additional elements, of which one is the Point of Beginning (POB), pertaining to IN service logic. A POB represents a starting point of service logic invoked from the Internet, which according to Figure 4.5 is a source internal to the service control space of the IIIN. Anything that takes place in the Internet before the invocation of IN service logic is not relevant to this model, which only includes the IN part of the IN. A SIB chain that starts with a POB ends internally to the service control space. Some IICSI1 SIBs contain Points of Extension (POEs), which are basically invocations of events in the Internet (that

extend the IN service logic). From the point of view of the IN, these invocations are entirely asynchronous and the IN gets no feedback as to the events the despatched message may have caused in the Internet.

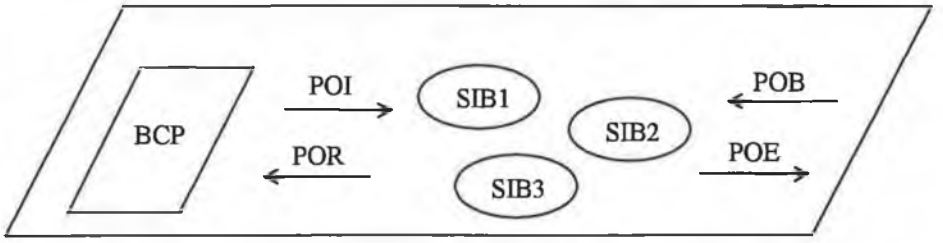


Figure 4.8 - Global Functional Plane Model for the IICSI1 IIIN

All the events that take place as parts of IICSI1 services are grouped into Internet Application Service Components (IASCs), but their functionality is not visible to the IN except through POBs and POEs. They are described in §4.3.4.2.

Since IIIN services include asynchronous invocations both from the telephone network and from the Internet, there can be two independently invoked SIB chains for the same service, and that is, in fact, how many of these services function. The independent lines of events, represented by independent SIB chains, are synchronised with the aid of the two newly defined SIBs, CALL INFORMATION UPDATE and RETRIEVE CALL INFORMATION, described in §4.3.3.3. The use of these SIBs is restricted to the case where information is passed from the SIB chain invoked from the IN to the SIB chain invoked from the Internet. This is because of the principle that the course of calls in the IN, once activated, must not be affected by events in the Internet (see §4.2.2.2.3).

The SIB chains representing the IICSI1 services, and using the SIBs described below, are shown in Appendix C.

4.3.3.2 sRelevant IN CS1 and CS2 SIBs

This section presents a list of CS-1 and CS-2 SIBs, which are needed to complete the IICSI1 functionality, and notes the actual services that make use of each listed SIB. More information on how exactly these SIBs are used by IICSI1 services can be found in Appendix C.

4.3.3.2.1 CS1 SIBs Used in IICSI1

Most IICSI1 services use the **USER INTERACTION** (UI) SIB, either for playing announcements or for data collection. Another SIB used a lot by the IICSI1 services is **SERVICE DATA MANAGEMENT** (SDM). The **TRANSLATE** SIB is used by services like Unified Messaging, Interactive Voice Response and Fax-on-Demand with a third party-provided special resource, where an abstract address representing a specific user of the special resource is translated into the actual address of the special resource. Finally, the **INCREMENT** SIB is used in the Tele-Voting service.

4.3.3.2.2 CS2 SIB Operations Used in IICSI1

As CS2 defines complex SIBs, consisting of atomic units called SIB operations and which are the equivalents of the CS1 SIBs, here the SIB Operations needed by the IICSI1 services are identified.

The End SIB Operation of the **END** SIB is used in a lot of the IICSI1 services to end the service logic initiated from the Internet. The Initiate service process SIB operation of the **INITIATE SERVICE PROCESS** SIB is used in the Fax-on-Demand service to start a separate branch of execution to send the fax (after the branch requesting the fax has returned).

4.3.3.3 The IICSI1 Service Independent Building Blocks

By analysing the IICIS1 services, listed in 4.3.2, it has been found that they require a group of additional SIBs to complement the capabilities already defined in IN. Those are:

CALL INFORMATION UPDATE (CIU),
NON-REAL-TIME VOICE-CHANNEL DATA MANIPULATION (NRT-VCDM),
RETREIVE CALL INFORMATION (RCI),
SEND SERVICE COMPONENT REQUEST (SSCR) and
TIMER.

The CIU SIB stores some call information together with an identifier (usually the calling line number) to make it accessible to other SIB chains of the same service instance, which do not have access to call information, primarily because they are initiated from the Internet and not from the SSF. The NRT-VCDM SIB controls the operation of the IN Special Resource Gateway, for the flow of information in both the directions (from the IN to the Internet and vice versa). The RCI SIB retrieves call

information posted by the CIU SIB. The use of a CIU – RCI SIB pair in two independent SIB chains (with no parent-child relationship between them) allows for inter-process communication between those SIB chains. The SSCR SIB sends a request for an Internet service component. The TIMER SIB makes the SIB chain sleep for a certain amount of time or until a specific moment, whichever is specified.

The complete Stage 1 descriptions of these SIBs, following the format used in ITU-T specification Q.1213, is given in Appendix A.

4.3.3.4 The IICSI1 Service-Dependent Building Blocks

Two services of the IICSI1, i.e. Automatic Call Distribution (ACD) and Group Announcements, need some highly service-dependent functionality, which will be described here. Each of these two Service-Dependent Building Blocks (SDBs), as they will be called, are in fact complex operations on particular service data and could no doubt be implemented as an appropriate sequence of SDM SIB instances and some other functionality. That, however, is not attempted here. The SDBs are inserted into the SIB chains in Appendix C in the same manner as the SIBs.

4.3.3.4.1 CALL LIST

This Service-Dependent Building Block finds the next free number from a list (passed to it) and calls it, returning after an answer has been received, with the exit point called GOTNUM. In the case of no answer, this SDB moves to the next number etc. It may also return if it reaches the end of the list, with two possible exit points: TRYAGAIN and ALLDONE. If it returns with TRYAGAIN, it means that the list has been exhausted but that not all the numbers have been successfully reached. If it returns with ALLDONE, all the numbers on the list have been called successfully. The SDB is designed to be called successively until the list is exhausted. It also creates a new list of numbers not successfully called in the current sequence, which is available on CALL LIST returning with TRYAGAIN. The new list can be used for a repeated sequence of calls to CALL LIST with the aim of reaching the previously unavailable numbers.

4.3.3.4.2 AUTOMATIC CALL DISTRIBUTION (ACD)

This SDB is somewhat similar to but simpler than CALL LIST. Each instance makes a call to the next free number on a list, returning when an answer is received.

The numbers are examined (for whether they are free) on a round-robin basis, whereas the CALL LIST SDB always starts searching from the beginning of the list.

4.3.4 Global Functional Plane – Internet Part

4.3.4.1 Introduction

For the Internet part of the IIN it is hard to identify well-defined service independent pieces of functionality, analogous to SIBs in that they have a beginning and an end in the same point of control, input and output parameters. However, the Internet-based processing and communication elements involved in the provision of IICSI1 services (described in 4.3.2) can be grouped into larger service components. The entire Internet part of the service for each IIN IICSI1 service, consists of one or more Internet Application Service Components (IASCs). IASCs are large chunks of functionality, classified depending on the type of interaction between the Internet and the IN that they involve. The IASCs are service-dependent and are not as finely granulated as SIBs.

From the point of view of the IN, an IASC is either an entity that causes a Point of Beginning (see §4.3.3.1) or an entity that is at the receiving end of a POE. An IASC can have either a synchronous or an asynchronous relationship with the IN. An IASC has an synchronous relationship with IN if it contains a wait for an event (message) from IN. Any IASC that does not wait at any moment for instructions from the IN has an asynchronous relationship with the IN.

Following are the descriptions of IASC types used with IICSI1 services. Which particular type of IASC is used for each of the IICSI1 services, and how they fit into the service SIB chains, can be seen in Appendix C. Each service contains one or two IASCs.

4.3.4.2 Internet Application Service Components (IASCs)

4.3.4.2.1 COLLECT DATA & SEND IN SERVICE REQUEST

This IASC type collects the data necessary to make a service request to the IN from an Internet user, then formats the data and sends it as a service request to the IN through the service control gateway. Once the service request is despatched to the IN, the IASC ends. Any errors that may occur from there on are reported to the user by other means. This type of IASC is used in the following services: Click-to-Dial,

Click-to-Dial-Back, Group Announcements, Tele-voting, Emergency Notification, Customer Profile Management, Alarm, Request to Fax and Request to Hear Content. IASCs of this type are invoked from the Internet and have an asynchronous relationship with the IN.

4.3.4.2.2 ROUTE INTERNET DATA WITH IN-SUPPLIED ADDRESS

This type of IASC routes some type of Internet-based data from one user on the Internet to another user on the Internet, while using address information acquired from the IN. The information is requested through a request for service causing a POB in the IN. The IN supplies the information in a POE. This IASC type is used in the Parallel Routing and Email by Number services. This is an IASC type with a synchronous relationship with the IN. It is invoked from the Internet.

4.3.4.2.3 PROCESS INTERNET SERVICE REQUEST

In this type of IASC a service request from the IN is processed i.e. the requested actions in the Internet are performed. In the IICSI1 set of services, there are only two Internet activities that can be requested from the IN. They are *pop-up on user's computer screen* and *send email message*. The services that use this type IASC are Click-to-Dial-Back, Group Announcements, Tele-voting and Automatic Call Distribution. Its relationship with the IN is asynchronous and it is invoked from the IN.

4.3.4.2.4 HANDLE SPECIAL RESOURCE DATA WITH IN-SUPPLIED ADDRESS

This type IASC comes into play in the functioning of a third-party provided special resource. It is used to get service instance information from the IN, in order to manipulate (most often send to an address in the Internet) voice-channel data received from the telephone network and translated into an Internet-native format. The necessary service instance information that is retrieved by this IASC (e.g. an e-mail address, found in the database using the dialled number) will have knowingly been retained by the IN when a call is made to the third party special resource in the telephone network. This IASC can be used in the following services: Unified Messaging, Fax on Demand and Subscriber-Specific IVR. This IASC type has a synchronous relationship with the IN. It is invoked by activities in the third party special resource from the Internet.

4.3.4.2.5 PROCESS INTERNET SERVICE REQUEST WITH SPECIAL RESOURCE DATA

In this type of IASC, voice channel data exchanged before or after this IASC through the IN special resource gateway, is manipulated appropriately in the Internet.

One case is the data is received through the special resource and converted in the gateway, then taken over by this IASC and sent to an appropriate address (e.g. some e-mail address) in the Internet. In another case this type of IASC fetches a document from the Internet, feeding it into the IN special resource gateway for further manipulation. The services that use this type of IASC are Unified Messaging, Fax-on-Demand, Subscriber-Specific IVR, Request-to-Fax and Request to Hear Content. IASCs of this type have an asynchronous relationship with the IN and are invoked from the IN.

This type of IASC is always associated with the NON-REAL-TIME VOICE-CHANNEL DATA MANIPULATION SIB, which contains a POE to invoke the IASC.

4.3.4.2.6 SEND SPECIAL RESOURCE DATA TO PSTN

This is an exceptional IASC type in that it may not really be involved with the IN. It represents the activity that takes place when the Request-to-Fax and Request to Hear Content are implemented with a third party-provided special resource gateway. A request is placed from the Internet for a document in the Internet to be fed into a third party-provided special resource gateway.

It is used in collaboration with IN activities in the Fax-on-Demand service, in the case that the fax to be sent is specified through user interaction in the IN. The IN then requests this IASC type to be executed, to send the specified document from the Internet to a specified address in the telephone network.

4.3.5 Distributed Functional Plane – IN Part

On the Distributed Functional Plane first the new Functional Entities introduced in IICSI1 and the already existing FEs which need modified functionality for IICSI1, are described; then the specification of the distributed functionality of the IICSI1 SIBs is presented.

4.3.5.1 IICSI1 Distributed Functional Plane Model

In Figure 4.9 the DFP Model, based on CS1 IN, is shown for IICSI1. As can be seen from the figure, the Functional Entities (FEs) introduced by the IICSI1 are the Internet-SCF Gateway Function (ISCGF) and the Internet-SRF Gateway Function (ISRGF).

The Internet-SCF Gateway handles the communication between the SCF and the Internet. It performs the translation between the various formats that the communication on the Internet may take (at the application level) and a restricted set of messages used by the SCF for communicating with the Internet. It also plays the role of a firewall protecting the IN from the wild world of the Internet, by checking the incoming service requests and filtering them appropriately.

The functionality of the IN special resource gateway is represented by the Internet–SRF Gateway Function (ISRGF). A third party SRF and a third party Internet-SRF Gateway Function (3pISRGF) have been added to the model for a complete picture, but in a separate domain, as they are not in fact part of the IN. The role of both the ISRGF and the 3pISRGF is to translate media flows, fax or speech, between the formats suitable for transport on the telephone network and those suitable for transport on the Internet.

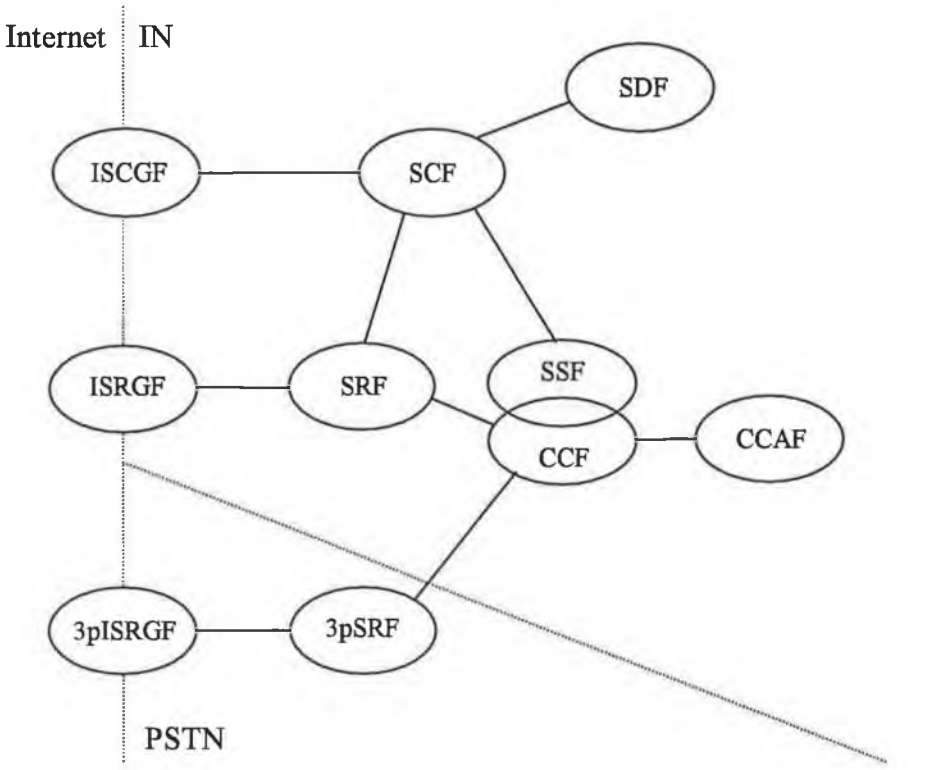
The IN functional entities discussed here are the SCF and the SRF, which need to be modified in order to be incorporated into the IIIN.

4.3.5.1.1 The SCF in IIIN

The SCF in the IIIN needs to be aware of the existence of the ISCGF. The Functional Entity Access Manager (FEAM) must allow for communication with the ISCGF. The SCF must have the capability to interpret messages from the ISCGF and to start Service Logic Programs (SLPs) on the basis of requests from the ISCGF, which will differ in format and parameters from those arriving from the SSF.

4.3.5.1.2 The SRF in IIIN

The IICSI1 set of services requires an SRF with some capabilities that are not present in the SRF of CS1. These are the capability to send and receive facsimile messages and the capability to receive and store voice messages. It must recognise SCF instructions related to these new capabilities.



CCAF - Call Control Agent Function
CCF - Call Control Function
SCF - Service Control Function
SDF - Service Data Function
SRF - Specialised Resource Function
SSF - Service Switching Function
ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
3pSRF – Third Party SRF
3pISRGF – Third Party ISRGF

Figure 4.9 - The Distributed Functional Plane Model for the IN Part of IICSI1 IIIN

4.3.5.1.3 Internet-SCF Gateway Function

The Internet-SCF Gateway Function (ISCGF) is a "thin" functional entity, with no service functionality of its own, which serves as the entry point for the SCF to the Internet and vice versa. From this description the ISCGF may seem entirely unnecessary, but it has been introduced for the following reasons:

- for security reasons the SCF should not be connected directly to the Internet; the security functions may be carried by an ordinary firewall, or a signalling gateway may be needed to translate between Internet-native communication mechanisms and the signalling mechanisms used in IN
- the existence of an intermediary between the SCF and other entities in the Internet allows for such intermediaries to be placed at different geographic locations to act as IN signal "collection/distribution stations" for requests coming from - and those being sent out onto - the Internet

- if the communication between the Internet and the IN makes use of different protocols and mechanisms in different instances of IN services, a functional entity separate from the SCF would relieve it from the additional and, irrelevant to IN services, functionality of interpreting and translating incoming and outgoing messages.

4.3.5.1.4 Internet-SRF Gateway Function

The Internet-SRF Gateway translates voice-channel data between telephone-network formats and Internet-native formats and thus serves as a bridge between an SRF in the IN and the Internet.

4.3.5.2 SIB Stage 2 Description

Stage 2 Descriptions of the SIBs introduced to cater for the needs of IICSI1 services can be found in Appendix B. The format of the Stage 2 Descriptions is the same as that used in ITU-T Recommendation Q.1214, and presented in Chapter 2.

For the Stage 2 SIB description any activities initiated in the gateways are not followed into the Internet beyond the gateways and will be considered as performed by the gateways themselves.

4.3.5.3 Relationships between FEs

With the new FEs, two new FE relationships are introduced into the Intelligent Network. One is the relationship between the SCF and the Internet-SCF Gateway (rIC), the other the relationship between the SRF and the Internet-SRF Gateway (rIR).

4.3.5.3.1 SCF-ISCGF

The rIC relationship consists of the following information flows:

Execute IN Service Component

a) FE relationship: ISCGF to SCF

b) Synopsis

This IF is issued by the ICG to request the execution of an IN Service Component i.e. of a Service Logic Program.

c) Information elements

ServiceComponentType	(mandatory)
ServiceComponentParameters	(optional)

d) IE description

ServiceComponentType identifies the SLP that is to be executed.

ServiceComponentParameters indicates the values of the set of SLP-specific input parameters for the SLP.

Execute Internet Application Service Component

a) FE relationship: SCF to ISCGF

b) Synopsis

This IF is issued by the SCF to request the execution of an Internet Application Service Component in the Internet.

c) Information elements

ServiceComponentType	(mandatory)
ServiceComponentParameters	(optional)

d) IE description

ServiceComponentType identifies the IASC that is to be executed.

ServiceComponentParameters indicates the values of the set of IASC-specific input parameters for the IASC.

Manipulate Internet Voice-Channel Data

a) FE relationship: SCF to ISCGF

b) Synopsis

This IF is issued by the SCF to request that data in the Internet, sent from the IN through the ISRGF, be sent to an address in the Internet or that data be fetched from an address in the Internet and fed into the ISRGF for the use of the SRF.

c) Information elements

OperationType	(mandatory)
InternetAddress	(mandatory)

d) IE description

OperationType identifies the operation that is to be performed, send or fetch.

InternetAddress indicates the address in the Internet (an e-mail address or a URL) that is to be used when sending or fetching.

One of the IFs, *Execute IN Service Component*, is not the part of any SIB. It is rather like the Initial DP IF sent from the SSF to the SCF. Also, the integration model for IIIN (see Figure 4.5), used in order to keep the IN distribution concepts as intact

as possible, commands that the effect of the *Execute IN Service Component* IF be regarded as an event generated by the SCF itself to trigger the execution of an SLP.

While the *Execute IN Service Component* and *Execute Internet Application Service Component* IFs operate within the integration models in Figure 4.5 and Figure 4.6, *Manipulate Internet Voice-Channel Data* is used in the context of the integration model in figure 4.7.

4.3.5.3.2 SRF-ISRGF

The rIR relationship, between the SRF and the ISRGF, is built on a different type of interaction to those seen between all the other Functional Entities. On this relation it is voice-channel data that is exchanged, in its raw form, in both directions.

In the case of a third party-provided special resource, the gateway and the special resource itself are not a part of IN and are not examined here.

4.3.5.3.3 SCF-SRF

The following is the IF added to the SCF-SRF relationship set of IFs:

Manipulate IN Voice-Channel Data

a) FE Relationship: SCF to SRF

b) Synopsis

This IF is issued by the SCF to request the voice-channel data manipulation by the SRF in co-operation with the ISRGF.

c) Information elements

CallPartyAddress	(mandatory)
OperationType	(optional)
UIParameters	(optional)

d) IE description

CallPartyAddress identifies the telephone number of the call party involved in the requested operation.

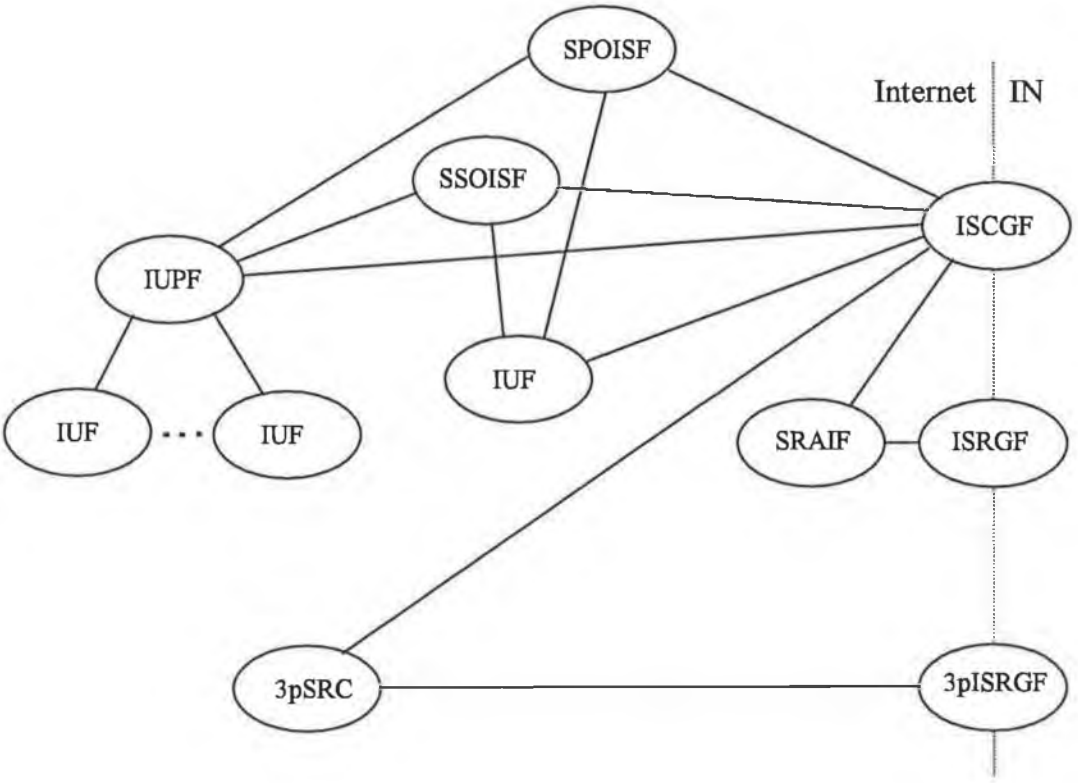
OperationType is the type of operation that is to be performed by the SRF in conjunction with the ISRG. FileType is the type that the file FileName has. This IE is used only in the case that several different file types are used to store service-relevant information. The operation type is one of the following (in relation to the Internet): send fax, receive fax, perform subscriber-specific user interaction and receive voice message.

UIParameters is an information element specifying the User Interaction SIB parameters in the case that UI needs to be performed as part of the operation. Such is the case of the Receive Voice Message operation, where a standard message may be played to the user before recording the user's message.

4.3.6 Distributed Functional Plane – Internet Part

4.3.6.1 Distributed Functional Plane Model

Figure 4.10 shows the DFP Model of the Internet part of the IIIN for IICSI1. Each of the entities, except the ISCGF and the ISRGF, can be contained in the IIIN several times. All these entities are described below, again excluding the ISCGF and ISRFG, since they have already been described in §4.3.5.



ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
3pISRGF – Third Party ISRGF
SRAIF – Special Resource Assisting Internet Function
3pSRC – Third Party Special Resource Controller
SSOISF – Service Subscriber-Owned Internet Server Function
SPOISF – Service Provider-Owned Internet Server Function
IUF – Internet User Function
IUPF – Internet User Proxy Function

Figure 4.10 - The Distributed Functional Plane Model for the Internet Part of IICSI1 IIIN

4.3.6.1.1 Internet User Function

The Internet User Function (IUF) is the IN-related functionality contained in an Internet user terminal. This functional entity provides the user with a communication-device type of interface. The IUF must have one or both of the following two capabilities, depending on the services that the IUF owner wishes to use:

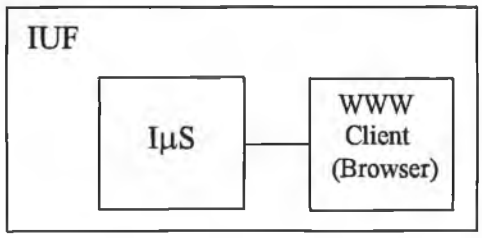
- service request submission;
- service request receipt and processing.

Service request submission is the capability to submit a request for an IIIN service to the ISCGF (and thus to the SCF). The actual IIIN-specific or service-specific mechanism for submitting a request could, theoretically, be contained in the IUF. However, the IIIN has been designed with the WWW, a freely exploitable and widely used system, in mind. If an IUF contains a WWW browser, it can use the browser to load the mechanisms for requesting different services, in the form of HTML forms linked to CGI scripts or Java Applets communicating with other entities in the Internet. The CGI script can be written to convey requests, received from the WWW client (through a submitted HTML form), to the ISCGF. A Java Applet could connect to the ISCGF directly and place a request for an IIIN service. In this way service providers not only provide a specific interface for each service, but can also add instance or service-specific data to the request that is being sent to the IN. It is obvious that one important component of an IUF is a WWW browser. Any other kind of IUF, e.g. one containing service-specific elements, is not considered here.

Service request receipt and processing is a capability that allows the IUF to receive requests and to act on them. With the aim of excluding any service-dependent functionality from the IUF, the functionality of receiving and processing requests has been placed into a minimal, daemon-like entity, which receives requests for some basic actions with parameters depending on the service. This entity has been called the IIIN Micro Server (IμS). For example, a request for a pop-up contains a reference to an HTML page, the IμS invokes the browser beside it in the IUF with that page reference as a parameter.

Another function of the IUF is to receive e-mail messages. It will not be examined here since any Internet user has some type of e-mail application in his/her computer. However, it is mentioned here for completeness, since some IIIN services *do* include the sending of e-mail messages to users.

The simple structure of the IUF is shown in Figure 4.11.



IUF – Internet User Function
IμS – Internet-Integrated Intelligent Network Micro Server

Figure 4.11 – IUF Model for IICSI1 IIIN

4.3.6.1.2 Internet User Proxy Function (IUPF)

The Internet User Proxy Function (IUPF) receives and distributes IIIN service request messages to the IUFs, while concealing their true identity from entities in the Internet, when an IUF or a group of IUFs are based behind a firewall for reasons of security. In this way the IUPF acts as a part of the firewall. The IUPF is not examined further in this work. However, even though it is not a functionally essential part of the IIIN, it would have to figure in any deployed instance of IIIN using IUFs behind firewalls.

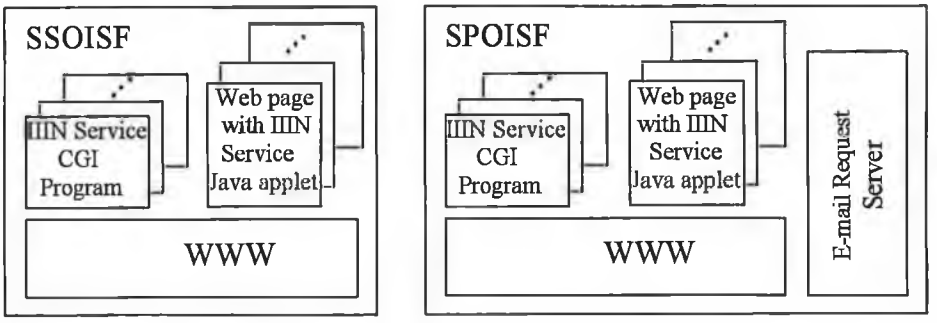
4.3.6.1.3 Service Subscriber-Owned Internet Server Function (SSOISF) and Service Provider-Owned Internet Server Function (SPOISF)

The Service Subscriber-Owned Internet Server Function (SSOISF) and the Service Provider-Owned Internet Server Function (SPOISF) have similar functionality. They both provide, to end users, data and mechanisms for requesting IIIN services. They differ in that the SPOISF provides services that are not associated with a subscriber and is owned by the network operator, much as the SCF or SDF, while an SSOISF provides the data and request mechanism for services requiring subscriber-specific elements and one can be owned by every subscriber. An example of the former is the Group Announcements service, while the Click-to-Dial-Back service is a typical example of the latter.

Both of these functional entities, on the one hand, make possible the "service–independent user's" access to IIIN services, by storing and offering, on a per-service-instance basis, the complex service interface that is one of the primary assets of the Internet in comparison with the IN. On the other hand, if it had not been for the

existence of the WWW, which provides that mechanism, the IIIN probably would not have been designed in that way. The coupling of the IUF with a WWW browser as its central component requires that both the SSOISF and SPOISF basically consist of a WWW server. All the other IIIN functionality is provided through specifically designed and implemented CGI scripts and Java Applets built into HTML pages.

An SPOISF, then, contains HTML pages with interfaces to such services as are not associated with a service subscriber. An example is the Customer Profile Management service feature, which can be used for the registration of IIIN users. One copy of a "service HTML page" is theoretically enough to provide the service. An SSOISF contains HTML pages with interfaces for services that do not have meaning unless they are associated with a service subscriber, more specifically, with some data pertaining to a service subscriber. The SSOISF must also provide that subscriber-specific data. An example of such a service is Parallel Routing, which cannot be used unless there is service subscriber-provided content to be sent to a user. For this type of service, a separate copy of the "service HTML page" must exist for each subscriber to the service.



a) SSOISF Model b) SPOISF Model

SSOISF – Service Subscriber-Owned Internet Server Function
SPOISF – Service Provider-Owned Internet Server Function
CGI – Common Gateway Interface
WWW – World Wide Web

Figure 4.12 – SSOISF and SPOISF Models for IICSI1 IIIN

Depending on how the service is implemented, the request placed by the user is sent to the IN (through the ISCGF) either through the SSOISF/SPOISF or directly. If the means of requesting a service is a Java Applet in an HTML page, loaded by the browser in the IUF, the applet may include the capability to communicate with the ISCGF and so the request for service is sent directly to the ISCGF. In the case that the

user is prompted to submit an IIIN service request in the shape of an HTML form, the request is first processed by a CGI program in the SSOISF/SPOISF and then sent to the ISCFG, by the SSOISF/SPOISF. Whether the capability of the SSOISF/SPOISF to communicate with the ISCGF is built into the CGI scripts or based in a separate entity, is a matter of implementation.The SPOISF has the additional capability of accepting service requests from the IN, through the ISCGF, needed by some services. The only service required by the SPOISF, in this context, for IICSI1 is the sending of e-mails with SCF-specified messages. Therefore, the SPOISF must, besides a WWW server, contain a server functionality that can accept and process requests for e-mail messages to be sent.

It should be mentioned that the SSOISF can be local or remote to the service user, where a local SSOISF is one that belongs to the same organisation/company as the IUFs using the service it provides. Subsequently, two types of IUFs, in an organisational (as opposed to functional) sense, can be identified: the subscriber IUF and the "free agent" IUF. Subscriber IUFs figure, for example, in the Parallel Routing service, as will be seen later in this chapter.

The SSOISF and SPOISF models are shown in Figure 4.12.

4.3.6.1.4 Third Party Special Resource Controller

A Third Party Special Resource Controller (3pSRC) is an application that controls a third-party-provided special resource. It is relevant to IN in that the SCF needs to be aware of the existence of the 3p SRC, in order to assist it in its operation. The dialled number for a third party-provided special resource is not its real address. The dialled number represents an Internet address and must be translated by the SCF twice: first into the address of the special resource, so that the call can be connected; then into the Internet address that it represents, which is stored temporarily in the SCF itself. Soon after the third party-provided special resource receives the call from the telephone network and before it proceeds with receiving voice-channel data, converting and sending it, the 3p SRC asks the SCF for the stored Internet address, and that is where it does finally send the data.

4.3.6.1.5 Special Resource Assisting Internet Function

The Special Resource Assisting Internet Function (SRAIF) is the entity to the Internet side of the ISRGF. It has two basic functions: one, to send to a specified address in the Internet data received from the ISRGF; two, to fetch data from the

Internet and to feed it into the ISRGF. These functions are performed following unidirectional requests from the SCF.

4.3.6.2 Distributed View of the IASC Types

4.3.6.2.1 COLLECT DATA & SEND IN SERVICE REQUEST

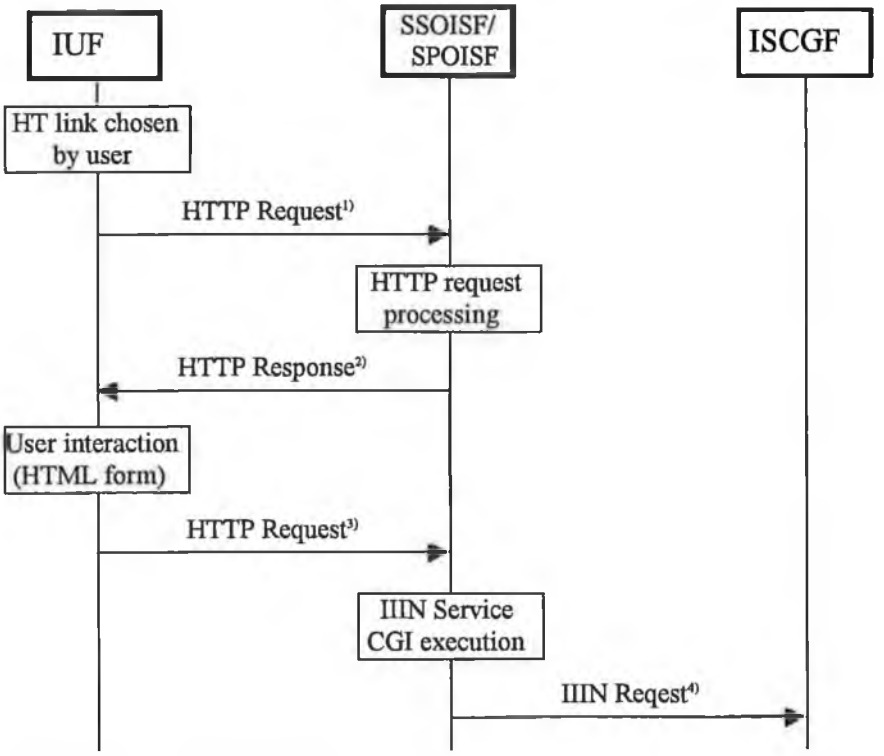
The two possible distribution scenarios for this type of IASC are shown in Figure 4.13, through Message Sequence Charts (MSCs).

In the case that the requests for service to the IN is shaped in an HTML form, the information contained in the form is returned to the SSOISF/SPOISF in an HTTP request, sent when the form is submitted. The form submission invokes a CGI script, which in turn sends a service request to the ISCGF.

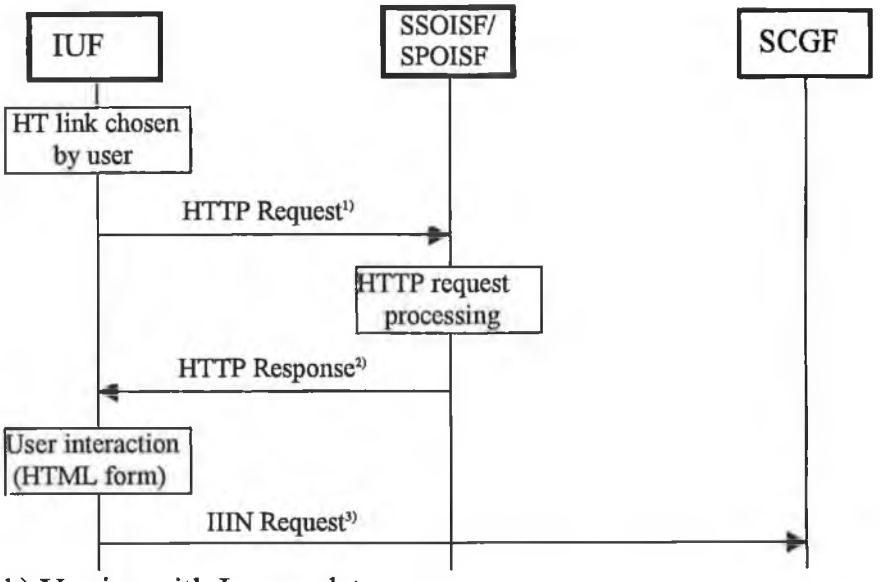
In the case that the request for IN service is submitted from an applet, the applet will have been built so as to contain communication functionality and thus the capability to place a service request directly towards the ISCGF.

4.3.6.2.2 ROUTE INTERNET DATA WITH IN-SUPPLIED ADDRESS

The most likely distribution scenario for this type of IASC is presented as an MSC in Figure 4.14. The user in the subscriber domain is presented with an HTML page containing the data that can be sent (routed) by this type of IASC (the subscriber IUF will have requested the page from the local SSOISF in the usual way of the WWW). For example, the HTML page may contain a sequence of pictures, each of which can be sent by the user, by "clicking" the mouse over the picture or over a Java applet to the side of the picture. Whatever the mechanism (CGI script or server waiting for requests from Java applets), a request to send data is sent to the SSOISF. The SSOISF processes the request then sends a request for service to the ISCGF and waits for a reply. Once the SCF has done the appropriate processing, it sends an address the ISCGF (in an *Execute Internet Application Service Component* Information Flow). The ISCGF conveys this IF in a return request to the SSOISF, which on receiving this message containing the address knows where to send the data originally submitted by the subscriber IUF. The destination should be the address of another IUF somewhere in the Internet and it is the IμS that actually receives the data, together with instruction as to what to do with it.



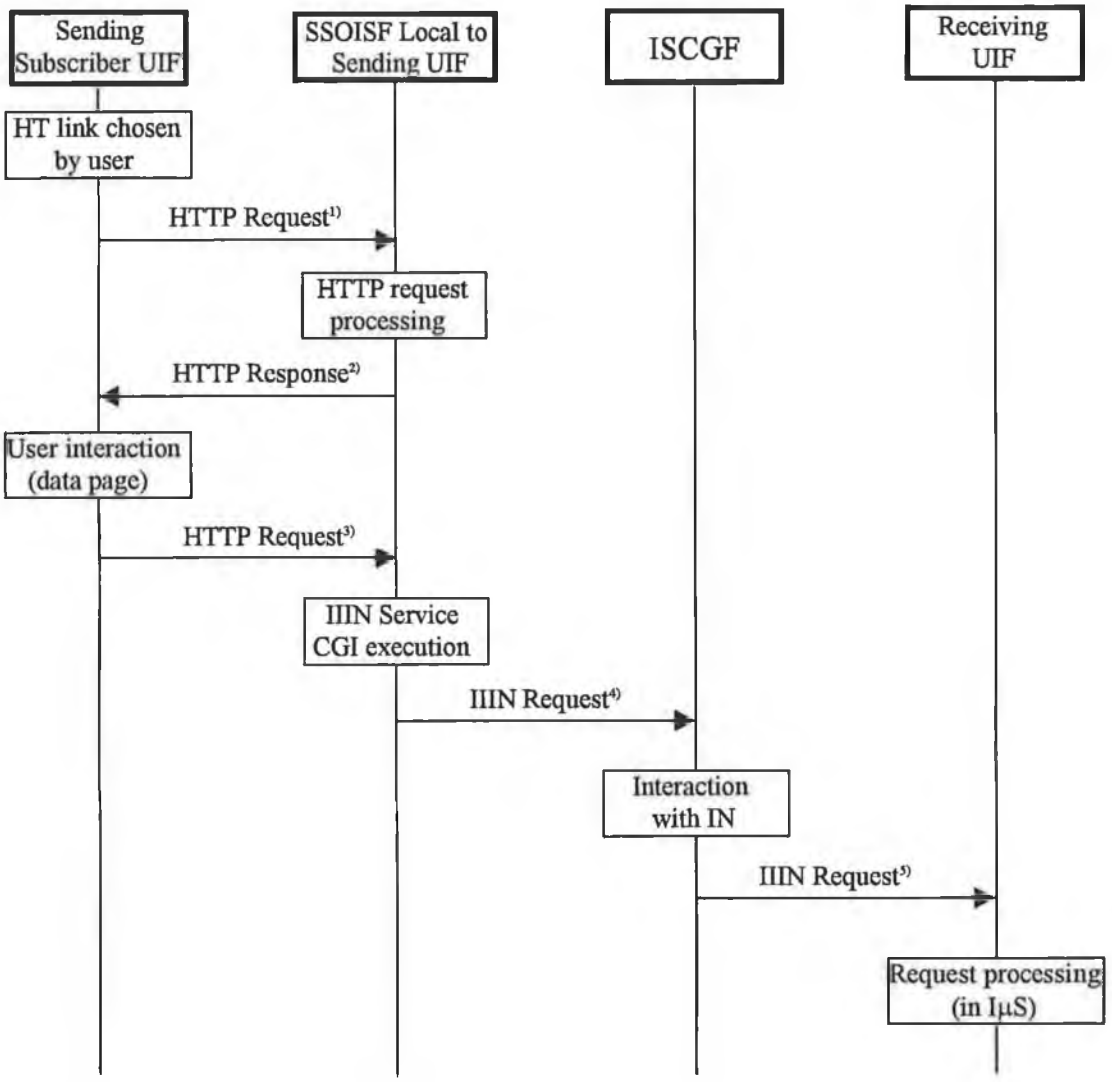
a) Version with CGI Program



b) Version with Java applet

IUF – Internet User Function
ISCGF – Internet-SCF Gateway Function
SSOISF/SPOISF – Service Subscriber/Provider-Owned Internet Server Function
NOTES: 1) Request for an HTML page containing either an IIIN Service HTML form or an IIIN Service Java applet
2) Response to the request described under 1)
3) Request submitting HTML form data
4) Request for IN Service Component, with format as described in Appendix B

Figure 4.13 – Information Flow Diagrams for the COLLECT DATA AND SEND IN SERVICE REQUEST IASC Type

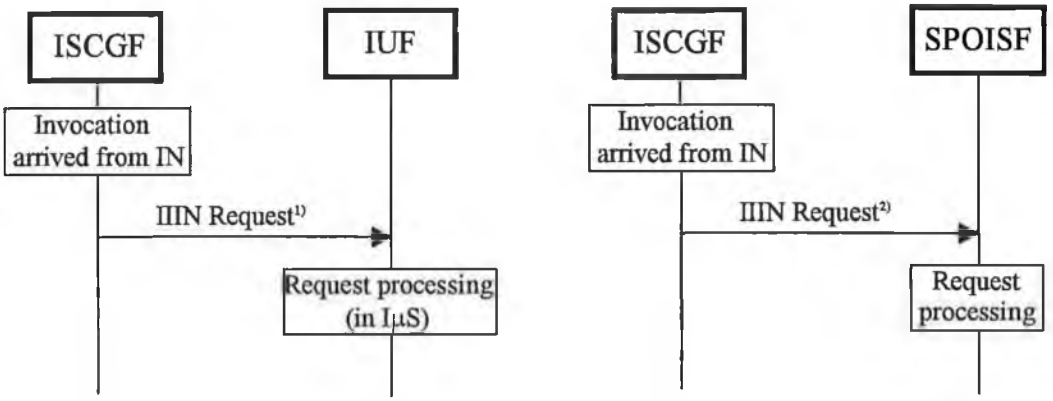


IUF – Internet User Function
ISCGF – Internet-SCF Gateway Function
SSOISF – Service Subscriber-Owned Internet Server Function
NOTES: 1) Request for an HTML page containing data to be sent
2) Response to the request described under 1)
3) Request for the execution of the IIIN Service CGI Program which will send the data (a reference to which is contained in this request)
4) Request for IN Service Component, with format as described in Appendix B
5) Request for Internet Service Component, with format as described in Appendix B

Figure 4.14 – Information Flow Diagram for the ROUTE INTERNET DATA WITH IN-SUPPLIED ADDRESS IASC Type

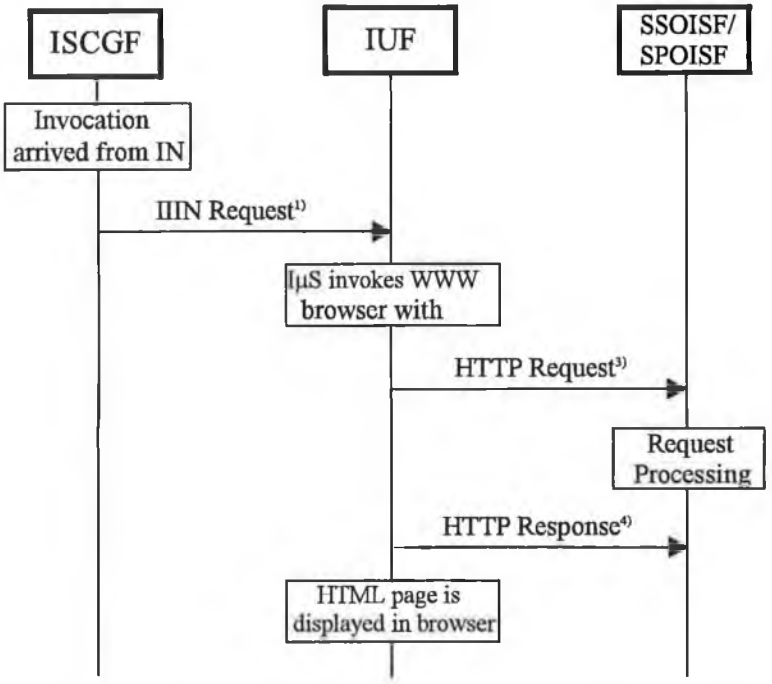
4.3.6.2.3 PROCESS INTERNET SERVICE REQUEST

The typical MSCs for this type of IASC are shown in Figure 4.15.



a) With Request to IUF

c) With Request to SPOISF



b) Screen Pop-Up IASC, as a concrete example of this IASC Type

IUF – Internet User Function
ISCGF – Internet-SCF Gateway Function
SSOISF/SPOISF – Service Subscriber/Provider-Owned Internet Server Function
NOTES: 1) Request for Internet Service Component, with format as described in Appendix B (an example of an Internet service component in the IUF is a screen pop-up)
2) Request for Internet Service Component, with format as described in Appendix B (an example of an Internet service component in the SPOISF is the sending of an e-mail message)
3) Request for Web page with data to be popped-up on the screen
4) Response to request described under 3)

Figure 4.15 – Information Flow Diagrams for the PROCESS INTERNET SERVICE REQUEST IASC Type

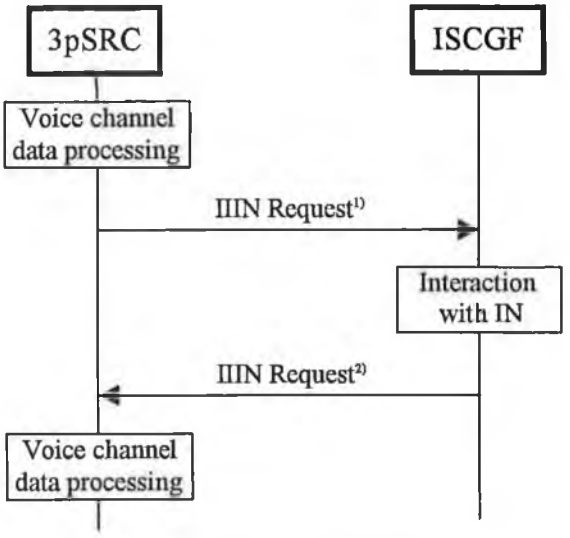
Figure 4.15a shows a scenario involving an IUF at the receiving end. The ISCGF receives the *Execute Internet Application Service Component* from the SCF

and sends a request for the execution of the component to the IμS. The IμS then processes the request. Figure 4.15b shows the MSC for the specific case of this IASC with a screen pop-up as the requested service component. After the IμS receives the request for the execution of a service component and deduces from the message that the requested component is a screen pop-up, it kicks off the browser with the reference to the document to be displayed as a parameter. Otherwise it may request from the browser to display that particular document, if the browser is already running.

Figure 4.15c shows another possibility for the distribution of this type of IASC. Here the ISCGF sends a request for service to the SPOISF. An example of an action requested out of the SPOISF is to send an e-mail message to a specified address.

4.3.6.2.4 HANDLE SPECIAL RESOURCE DATA WITH IN-SUPPLIED ADDRESS

The MSCs for this IASC type are shown in Figure 4.16.



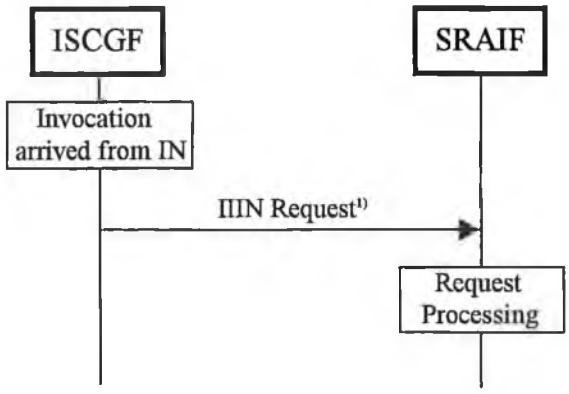
ISCGF – Internet-SCF Gateway Function
3pSRC – Third Party Special Resource Controller
NOTES: 1) Request for Internet Service Component, with format as described in Appendix B, which will return an Internet address in another IIN Request
2) Request containing an Internet address, sent in response to 1), with format as described in Appendix B

Figure 4.16 – Information Flow Diagram for the HANDLE SPECIAL RESOURCE DATA WITH IN-SUPPLIED ADDRESS IASC Type

Here a service request is sent directly from the 3pSRC, which has received (or is in the process of receiving) voice-channel data from the telephone network hitherto without IN communication, to the ISCGF, which conveys the request in the form of an *Execute IN Service Component* IF to the SCF. The SCF sends the address back in an *Execute Internet Application Service Component* IF to the ISCGF. The ISCGF sends the address to the waiting 3pSRC. The 3pSRC can then use the address to send the received data in the Internet.

4.3.6.2.5 PROCESS INTERNET SERVICE REQUEST WITH SPECIAL RESOURCE DATA

In this type of IASC the ISCGF sends a request for service to the SRAIF. The request is either for the despatch of voice-channel data, received through the ISRGF, to an Internet address (e.g. an email address) or for the retrieval of data from an Internet address (e.g. a URL) which is to be fed into the ISRGF for format conversion. The MSC for this type of IASC is shown in Figure 4.17.

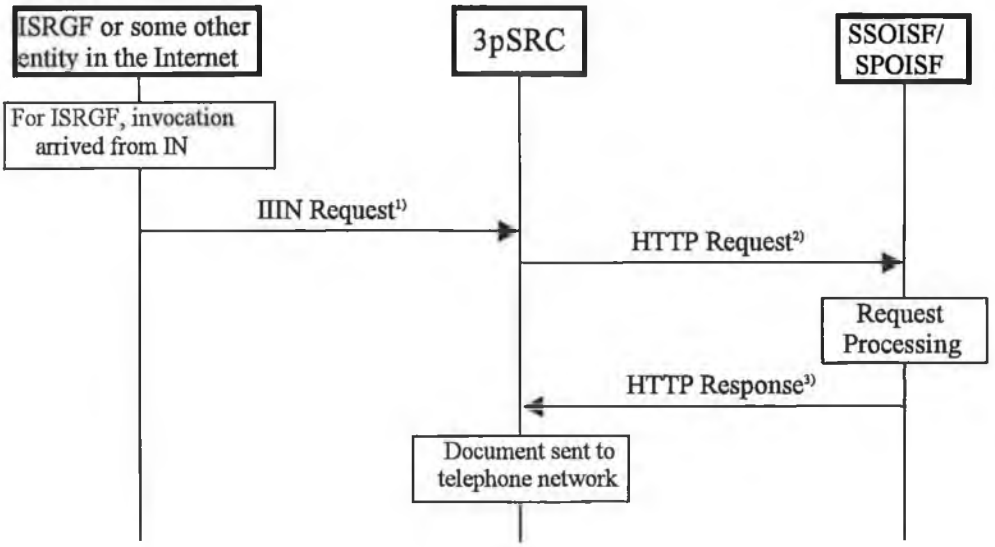


ISCGF – Internet-SCF Gateway Function
SRAIF – Special Resource Assisting Internet Function
NOTES: 1) Request for Internet Service Component, with format as described in Appendix B

Figure 4.17 – Information Flow Diagram for the PROCESS SERVICE REQUEST WITH SPECIAL RESOURCE DATA IASC Type

4.3.6.2.6 SEND SPECIAL RESOURCE DATA TO PSTN

In this IASC type, the ICSGF or an entity in the Internet requests from the 3pSRC to fetch a document from the Internet and to send it through the ISRGF to a specified address in the telephone network. The MSC for this IASC type is shown in Figure 4.18.



ISCGF – Internet-SCF Gateway Function
3pSRC – Third Party Special Resource Controller
SSOISF/SPOISF – Service Subscriber-Owned Internet Server Function
NOTES: 1) Request for Internet Service Component, with format as described in Appendix B
2) Request for document to be sent (as fax or spoken content) to the telephone network
3) Response containing document requested under 2)

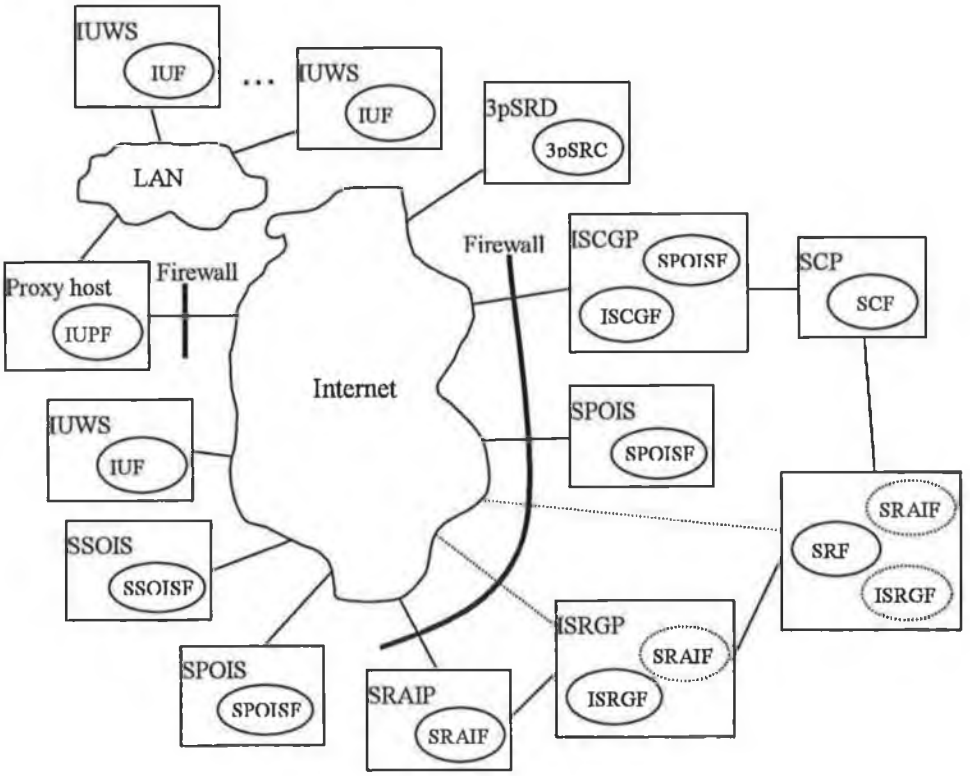
Figure 4.18 – Information Flow Diagram for the SEND SPECIAL RESOURCE DATA TO PSTN IASC Type

4.3.7 Physical Plane

4.3.7.1 FE-PE Mappings

The physical entities of the IICSI1 IIIN are shown in Figure 4.19, and so are the various mappings of the IIIN Functional Entities (FEs) onto those physical entities. Figure 4.19 also shows the physical entities of the exiting IN that are affected by the additional functionality of the IIIN. The mappings of FEs onto PEs are summarised separately in Table 4.1.

With regard to its direct exposure to the Internet, the ISCGF must, for security reasons, be based in a PE separately from all the other FEs of the Intelligent Network. This PE is the Internet-SCP Gateway Point (ISCGP). Communication between the ISCGP and the SCP could use either the SS7 signalling protocol or TCP/IP. Either way, the communication takes place within the protected IN network and the ISCGP contains the firewall, which is part of the ISCGF.



Physical entities:
SCP – Service Control Point
IP – Intelligent Peripheral
ISCGP – Internet-SCF Gateway Point
ISRGP – Internet-SRF Gateway Point
SPOIS/SSOIS – Service Provider/Subscriber-Owned Internet Server
SRAIP – Special Resource Assisting Internet Point
3pSRD – Third Party Special Resource Device
IUWS – Internet User Workstation

Functional entities:
SCF – Service Control Function
SRF – Special Resource Function
ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
SPOISF/SSOISF – Service Provider/Subscriber-Owned Internet Server Function
SRAIF – Special Resource Assisting Internet Function
3pSRC – Third Party Special Resource Controller
IUF – Internet User Function
IUPF – Internet User Proxy Function

Figure 4.19 – Mapping of Functional Entities onto Physical Entities for IICSI1 IIIN

The ISRGF may be based in the IP or on a separate physical entity called the Internet-Special Resource Gateway Point (ISRGP). The Special Resource Assisting Internet Function could be in a separate physical entity (Special Resource Assisting Internet Point – SRAIP) or based together with the ISRGF on the ISRGP. Finally, it could be based on the IP, but only in the case that the ISRGF is also located on the IP.

Since the SRAIP uses standard Internet Applications (WWW and e-mail) to communicate with the Internet, it does not create any security problems as long as it is based within the IN network, protected by a standard firewall.

The SSOISFs and SPOISFs are all mapped onto separate physical entities called Service Subscriber-Owned Internet Servers (SSOISs) and Service Provider-Owned Internet Servers (SPOISs), respectively. The SSOISs are based anywhere in the Internet, outside the IN network. Some SPOISs may, however, be based within the protected IN portion of the Internet, and even in the same physical entity as the ISCGF, the ISCGP.

The IUPF is based in the proxy of the subscriber's private network, while the IUFs are based in Internet User Work Stations (IUWSs), e.g. PCs.

The location of the 3pSRC is on a third party-provided Special Resource Device
(3pSRD), connected to the Internet.

Functional Entity	Possible Physical Entities
ISCGF	ISCGP
SPOISF	ISCGP, SPOIS (either behind the network operator's firewall or not)
SSOISF	SSOIS
ISRGF	ISRGP, IP
SRAIF	SRAIP, ISRGP (possible only if the ISRGF is based on an ISRGP), IP (possible only if the ISCGF is based on the IP)
IUF	IUWS
IUPF	Internet Proxy Host
3Psrd	3pSRC

Table 4.1 – Mapping of Functional Entities onto Physical Entities for IICSI1 IIIN

4.3.7.2 Communication Between IIIN Entities in the Internet

On the Internet, all communication takes place through TCP/IP or through higher level protocols that use TCP/IP for transport. The communications occurring as part of the IASCs are no exception.

The protocols suggested for use in the Internet part of the IIIN are the Hypertext Transfer Protocol (HTTP), the Simple Mail Transfer Protocol (SMTP), the

Internet Inter-ORB Protocol (IIOP) or some other CORBA protocol and a protocol called the IIIN Text-Based Protocol (IIINTBP), which has been defined for this project.

HTTP is an application-level protocol which uses TCP/IP for the transport of data. It is text-based, and so implies the transfer of data in the form of character sequences representing text. HTTP is used in the IIIN exclusively within the context of WWW browser – server communication, except in one case (between the ISCGF and the IUPF), which is described below.

SMTP is also a text-based protocol and it is used for the exchange of email messages, both in general and in the context of this project.

The CORBA protocols are used for the exchange of requests and replies between CORBA clients and servers. Within IIIN (as in most other cases), these protocols use TCP/IP as a means of transporting data over the Internet.

Finally, the IIINTBP uses TCP/IP for the transport of data, and transfers data in text-format, similarly to HTTP and SMTP.

Where these protocols are used in the PE to PE relationships is illustrated in Figure 4.20 and explained in the following subsections.

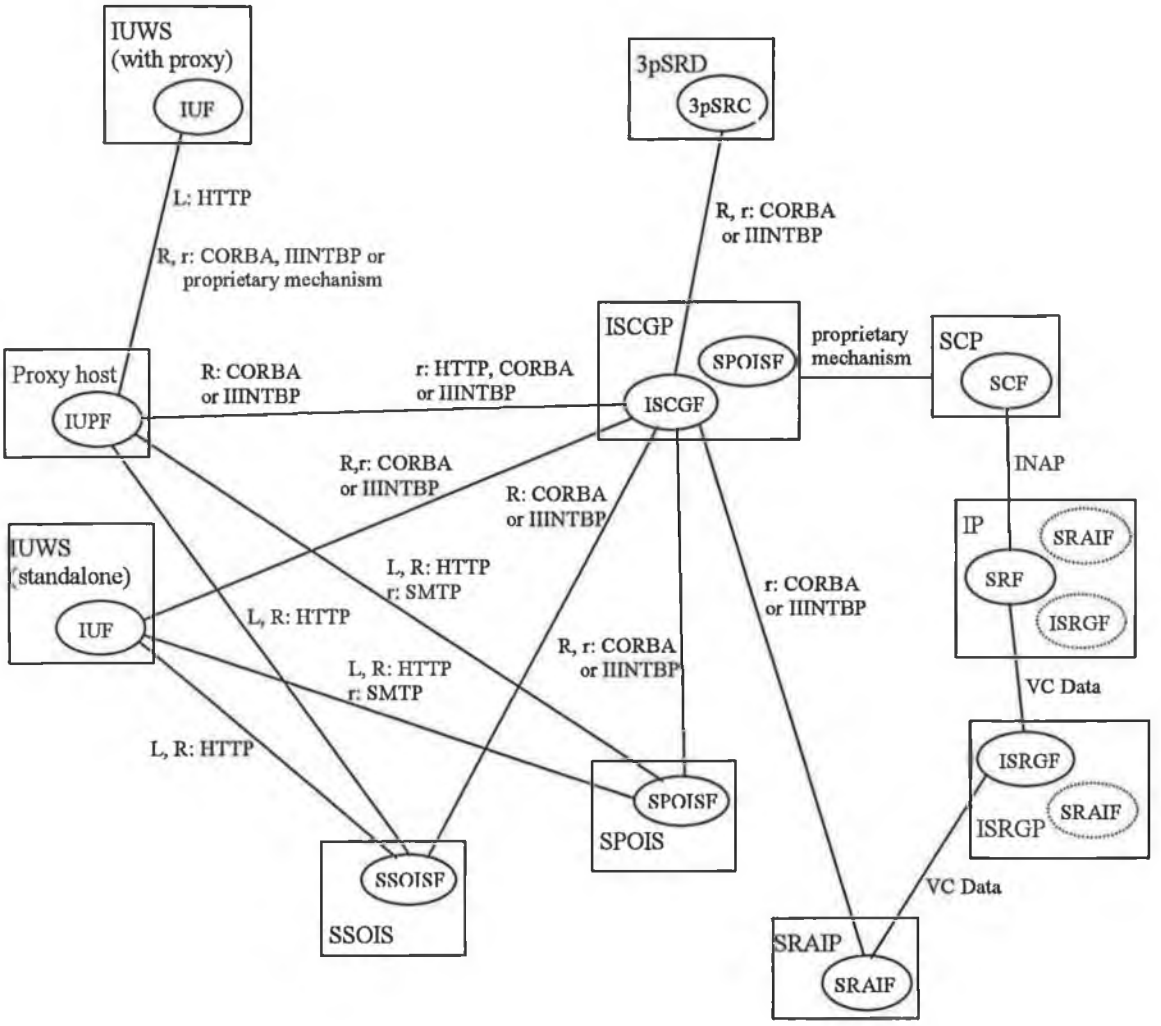
4.3.7.2.1 IUWS-SPOIS/SSOIS

The purpose of the link between the IUWS and an SPOIS/SSOIS is the provision, from the SPOISF/SSOISF to the IUF, of data and mechanisms for placing IIIN service requests. Since these activities are implemented as WWW activities, the relationship is based on the HTTP protocol. This is true also as regards the relaying of IN service requests from the IUF, by the SPOISF/SSOISF. In that case the first leg of the IN service request's journey, from the IUF to the SPOISF/SSOISF, is an HTTP-based request carrying data submitted through an HTML form.

Apart from that, the SPOIS communicates with the IUWS by sending it email messages on behalf of the SCF, when prompted to do so by a message from the ISCGF. For that purpose, SMTP is used.

4.3.7.2.2 IUWS–ISCGP

Communication between the IUWS and the ISCGP takes place within one of two different contexts.



Physical entities:
SCP – Service Control Point
IP – Intelligent Peripheral
ISCGP – Internet-SCF Gateway Point
ISRGP – Internet-SRF Gateway Point
SPOIS/SSOIS – Service Provider/Subscriber-Owned Internet Server
SRAIP – Special Resource Assisting Internet Point
3pSRD – Third Party Special Resource Device
IUWS – Internet User Workstation

Functional entities:
SCF – Service Control Function
SRF – Special Resource Function
ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
SPOISF/SSOISF – Service Provider/Subscriber-Owned Internet Server Function
SRAIF – Special Resource Assisting Internet Function
3pSRC – Third Party Special Resource Controller
IUF – Internet User Function
IUPF – Internet User Proxy Function

HTTP – Hypertext Transfer Protocol
IIINTBP – Internet-Integrated Intelligent Network Text-Based Protocol
VC Data – Voice Channel Data

Types of relationships:
L: denotes an ordinary WWW browser – server relationship, used here for the loading of the service interface into the user's browser
R: denotes a service request from the Internet towards the IN part of the system
r: denotes a service request in the direction from the IN towards some entity in the Internet

Figure 4.20 – Mapping of Protocols onto Relationships Between Physical Entities in the IIIN

First, the IUF may send a service request to the ISCGF, using a mechanism provided by the SPOISF/SSOISF as a Java applet in an HTML page. The Java applet places a request to the ISCGF using a CORBA method invocation over a CORBA protocol, such as IIOP, the IIINTB, or, possibly, using the Java Remote Method Invocation (RMI) mechanism (which is a Java – specific mechanism similar to CORBA).

In the other context, the ISCGF sends a request to the IμS in the IUWS and this can be done either in the form of a CORBA method invocation or using the IIINTBP.

4.3.7.2.3 SSOIS/SPOIS-ISCGP

Communication between the SSOIS/SPOIS and the ISCGP occurs if the SSOISF/SPOISF is relaying an IIIN service request from the IUF. In this scenario, a CGI script in the SSOISF/SPOISF sends an IIIN service request to the ISCGF. This could be done in the form of a CORBA method invocation, or using IIINTBP. Indeed, if the SPOISF and the ISCGF are located in the same physical entity, some internal communication mechanism can be used.

The other form of communication between the ISCGP and the SPOIS, takes place when the ISCGF requests of the SPOISF to send an email to a user (in fact, an email program in the IUWS). This communication takes the form of either CORBA invocations or IIINTBP messages.

4.3.7.2.4 3pSRD–ISCGP

The communication on this relation happens when the 3pSRC needs to obtain Internet routing information from the SCF. Similarly to the above case, CORBA invocations or the IIINTBP can be used.

4.3.7.2.5 ISCGP-SRAIP

On this relation, requests are sent by the ISCGF to the SRAIF, for the processing of voice-channel data. The requests may be CORBA invocations or IIINTBP messages.

4.3.7.2.6 SRAIP-ISRGP and ISRGP-IP

If these relations exist in the system (i.e. the separate physical entities exist), some type of data-transfer protocol is used, since on these relations large amounts of voice-channel data are transferred.

4.3.7.3 Protocol Definition

The messages used between the ISCGF and other Functional Entities in the Internet should be standardised, in order to allow easy incorporation of new SSOISFs, SPOISFs, IUFs, 3pSRCs and SRAIFs into the system. There are only three information flows that the ISCGF needs to send or receive. A suggestion for a standard format of these three messages in two different forms, CORBA IDL and the text-based IIINTBP, is given in Appendix D.

The messaging between the ISCGF and the SCF need not be standardised, since it takes place on a relation internal to the IN and not accessible by any entities except those two.

The standardisation of the data transfer methods between the SRAIP and the ISRGP or between the ISRGP and IP is not of interest for this project.

4.4 IIIN Deployment Considerations

The previous sections of this chapter have given a description of the IIIN without considerations for how it may be deployed in a real-life environment and the issues that such deployment might raise. Here is a short account of those issues.

4.4.1 Security

One of the important things that need to be tackled in IIIN is security. The security of a system has two aspects: secure transmission of data between entities of the system and restriction of access to the system so as to allow access at different levels to users or operators with different levels of authority. In the IIIN security problems arise at the point of connection of the IN part of the IIIN with the Internet part, since IN in itself is a closed system and implicitly secure.

Secure transmission in the Internet can be achieved by the use of private and public key encoding or some other encoding mechanism. Where restricted access to IIIN entities is concerned, firewall technology should be used [Oppl97].

If the communication is between trusted entities, such as the ISCGF (representing the SCF) on one side and an SPOISF/SSOISF or an 3pSRC on the other, filtering of incoming packets at the IP layer, on the basis of the source IP address, could be done. If the incoming data to the ISCGF (SCF) is from an non-trusted entity, such as an IUF, the filtering of data should be at the application level, on the basis of application type, message format and the like. Between the SRAIF and the ISRGF, all incoming data must be from trusted entities (service subscribers).

4.4.2 Performance

Another issue that arises in the IIIN from the fact that it incorporates the Internet is performance. Unlike the telephone network (voice-channel and signalling), the Internet is an unmanaged network, with no guarantee of quality of service. Therefore, the Internet as it is today cannot be used for high quality real-time services and that is the reason why the choice of services to be included in IICSI1 services has been limited to those not critically depending on real-time performance of the Internet.

However, as the Internet protocols develop in the direction of providing quality real-time services, with IPv6 (Internet Protocol version 6) at the network layer, the universal provision of services with a more involved integration between the IN and the Internet should become possible in the near future. Examples of such services are voice-over-IP and call-control functionality based in the Internet e.g. graphical call-party manipulation.

The IICSI1 architecture for the IIIN suggests the use of Internet-SCF Gateways (ISCGFs) distributed geographically and communicating with the SCF through the SS7 network. Thus the distance covered by the IIIN communication messages in the Internet is reduced, which should slightly improve real-time performance of the IIIN.

4.4.3 Registration

Another point that needs to be considered in relation to deployment is the registration of users to access IIIN services. Registration here means the logical association of the telephone number of the IN user with the IP address of the

computer belonging to same user, the telephone and the computer lying physically beside each other and accessible to the user at the same time. The act of registration would also automatically allow the user to obtain IIIN Internet host communication software (the IμS).

The association between the telephone number of the IN user and the IP address of the computer belonging to the same user can be static or dynamic. In the case of a static association, the association is formed on a long-term basis and is applied to any IIIN service instance that takes place while it is in effect. A dynamic association is formed on a per-service-instance basis and is dissolved as soon as the service instance has been entirely executed.

As regards IIIN services, some of them require a static association in order to function at all. An example of such a service is Parallel Routing, which requires the association for the receiving user to be static. Other services can be implemented in either way, that is using either static or dynamic associations. An example of such a service is Click-to-Dial.

4.4.3.1 Static Association

A static association between the telephone number and IP address for a user would have to be recorded in the Service Data Function of the Intelligent Network. This could be done through a specially designed instance of the Customer Profile Management feature, with the user accessing the feature either through a telephone in the IN or through a computer on the Internet. The actual functionality of the feature would vary considerably depending on the type of connection used for communication to the Internet.

In the case of a modem connection, or the use of the Dynamic Host Configuration Protocol (DHCP) on a LAN, where IP addresses are assigned temporarily on a connection-to-connection basis, an association would have to be formed each time the IP address changes. The user might have an IIIN registration in place on a long-term basis that prompts the recording of a new association in the SDF each time a modem connection is established.

In the case of a permanent connection of an IP host to the Internet, it is most likely that the access to all available Internet services of such a host will be through a proxy server. In that case the proxy server would have to be registered as taking part in the IIIN, using a proxy registration CPM feature. Each host using that proxy would have to be registered specially as taking part in the IIIN through the proxy in

question. The actual communication within IIIN services with such hosts would take place through the proxy and would use a different mechanism from the communication with hosts directly connected to the Internet. The group of users using the proxy would have to obtain IIIN proxy communication software as well as the IIIN Internet host communication software (the IμS), which would become available to them at the time of registration of the proxy.

4.4.3.2 Dynamic Association

Dynamic association functions in a much simpler manner than static association but is limiting as to the types of services that can be implemented using it. Dynamic association requires the user's telephone number to be stored in the user's computer and to be sent as part of any IIIN request from that computer. This method is, obviously, restricted to services that involve IIIN requests originated only from the Internet side of the IIIN, whether those requests are of the type that kicks off the service or are intermediate requests, prompted by some other IIIN service activity.

In the case of the implementation suggested and described in this work, the user's telephone number would have to be stored within the WWW browser, as a configurable Java class or in some other way. Then each time an IIIN request is placed from that browser, the telephone number is included in the data sent with the request to the Internet-SCF Gateway (or to the SSOISF/SPOISF).

4.4.4 Billing

Billing in the IIIN would have to be based on the billing system already available in the associated IN. Users would most likely be billed according to the usage of the Intelligent Network part of the IIIN and not the Internet, also depending on the type of service involved. The users would be billed though their usual telephone bills, e.g. monthly.

For example, in the case of the Parallel Routing service, the IN is involved in providing the data needed to route a message through the Internet, even though this may not be obvious to the user. Depending on the policy implemented by the service provider, the user could be billed on a per-call basis (for each Parallel Routing performed) or through subscription, in which case the subscription fee must be paid in advance for the Parallel Routing to be performed at all. The type of billing used, of course, affects the way the service is implemented.

4.4.5 Simultaneous Availability of Internet/Telephone Connection to the User

The probably biggest drawback of the proposed system is the way that private users most commonly access the Internet – through a modem connection. The use of a modem means a busy telephone line and the impossibility of simultaneous use of the telephone. Users with modem connections to the Internet would be able to use only a subset of IIIN services: those that do not involve the simultaneous use of the Internet and voice connections.

On the other hand, there are a large number of Internet users that do have the possibility of simultaneous access to the telephone network. Those are organisations and companies with direct connections to the Internet, private users with more than one telephone line and users with special connections that combine telephone and Internet traffic, such as with AVD, SVD or DSVD connections [Hoog96].

5 The IIIN Prototype Implementation

5.1 Introduction

This chapter offers a description of the prototype of the Internet-Integrated Intelligent Network (IIIN) that has been built as part of the project. The prototype has been implemented together with six sample services, which demonstrate its functionality. Two of the services, Abbreviated Dialling and Call Forwarding, are "pure" Intelligent Network (IN) services, provided by the basic IN. The other four services, Customer Profile Management (for the registration of an IIIN user), Click-to-Dial-Back, Group Announcements and Parallel Routing, are IIIN services, that is, they are provided by the combined system comprising of the basic IN and the Internet. It should be noted that the prototyping of an IIIN containing an Internet-SRF Gateway is beyond the scope of this project. Therefore, the IIIN prototype that has been built contains only an Internet-SCF gateway and supports only those services that do not require an Internet-SRF Gateway.

The prototype was built in two stages, to reflect a potential real-life IIIN development and deployment situation where, quite naturally, the IIIN would be built to extend an existing IN network. In the first stage a prototype of the basic IN was built, and the two "pure" IN sample services were implemented to use that platform. In the second stage the Internet component was added to the basic IN prototype, and thus the prototype of the IIIN was completed. The four IIIN sample services were implemented to run on that prototype of a full IIIN platform.

5.2 Equipment and Tools

Following is a description of the equipment and software tools used for the development of the Internet Integrated Intelligent Network (IIIN) prototype in this project.

For its switching functionality the IIIN platform prototype relies on a PCX - 512 host programmable digital switching platform, supplied by Excel. Of the several components that can be contained in such a platform, this project uses three: the MX/CPU-512, which contains the central processor and switching matrix for 64Kb voice channels; the PCULC-12, which is an adapter for the integration of 12 analogue

lines into the PCX-512 system; and the PCMFDSP card, which performs several Digital Signal Processing functions, of which the DTMF receiver is used in this project.

The software supplied with the PCX-512 consists exclusively of hardware controllers, whilst its behaviour is fully defined and controlled from a host. The communication between the host and the PCX-512 takes place over an RS-232 link and is based on a primitive message set proprietary to Excel.

The host used in this project to control the PCX-512 is a PC containing a Pentium processor and running Windows NT 4.0. The software for communicating with the PCX-512 and that of the basic IN platform prototype run on the same PC and have been developed using Borland C++.

The IIIN extension to the IN has been built to run on UNIX, specifically on SUN Sparc Workstations running Solaris (different versions). The development was done in C++, using the SUN WorkShop CC compiler.

The Java Development environment and compiler used was JDK1.1.6, from SUN Microsystems.

The Internet is represented by the office LAN running TCP/IP and containing a Netscape-Enterprise/3.5.1 WWW server. The browser used is Netscape. The IDL translator and CORBA environment used are those supplied by Visigenic.

Four analogue phones (connected to the PCULC-12 card in the PCX) and several PCs were used as end-user equipment for testing the prototype-carried IIIN services.

5.3 The Basic IN

5.3.1 Platform Prototype

5.3.1.1 Implementation Model

The entire basic IN platform prototype runs on one NT-based Pentium PC. Its implementation model is shown in Figure 5.1. All the modules presented here have been implemented using the C++ programming language and compiled using the Borland C++ compiler. They are all in Windows NT executable format and run on a single Windows NT 4.0-based PC.

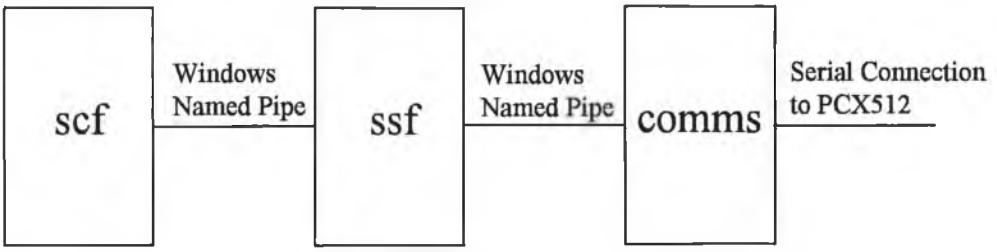


Figure 5.1 – Implementation Model for the IN Platform Prototype

There is a communications module, *comms*, which performs the basic communication functions for the serial port towards the PCX-512 switch.

The Call Control Function/Service Switching Function (CCF/SSF) as well as the Call Control Access Function (CCAF) are contained in module *ssf*. This module communicates with the *comms* module using the Windows Named Pipes mechanism. In the exchange of messages between the *ssf* module and the PCX-512 switch, which takes place with the *comms* module as an intermediary, the CCF exercises control over the PCX switching functions, while the CCAF controls the analogue lines connected to the switch.

The *scf* module contains the Service Control Function (SCF) and the Service Data Function (SDF). This module communicates with the *ssf* through Windows Named Pipes.

5.3.1.2 Object Models

5.3.1.2.1 ssf Module Object Model

The high-level UML class diagram for the *ssf* module is shown in Figure 5.2. The PipeReaderWriter reads from and writes to the Windows Pipes connecting the *ssf* module to the *comms* module and the *scf*. The CCAF interprets received messages from the switch and formats messages to be sent to the switch, which pertain to the behaviour of the phones connected to the switch, as well as those pertaining to the behaviour of the switching matrix. (Control of the switching matrix is, indeed, not part of the functionality of the CCAF Functional Entity but, dictated by the nature of the host-programmable switch, an object like this CCAF made for simplest implementation.) The CCAF object contains the address configuration of the phones

connected to the switch[17]. The BCM controls the creation and destruction of Basic Call State Model objects as calls are initiated and ended. It also performs call-unrelated functions of the switch (such as call filtering). The BCSM class is the base class for the O_BCSM and T_BCSM, the Originating and the Terminating BCSMs. The BCSMs contain the basic call control functionality for the originating and terminating parts of a call. This functionality is punctuated by Detection Points (DPs), at which criteria for the interruption of the call (in order to send a service request to the SCF) are examined. The initial and further call control exercised respectively by the BCM and the BCSM, takes place through the CCAF The DPManager is a helper class to the BCSM. It keeps track of the dynamic and static arming and disarming of DPs. The SCFAM formats SCF-bound messages and interprets messages received form the SCF.

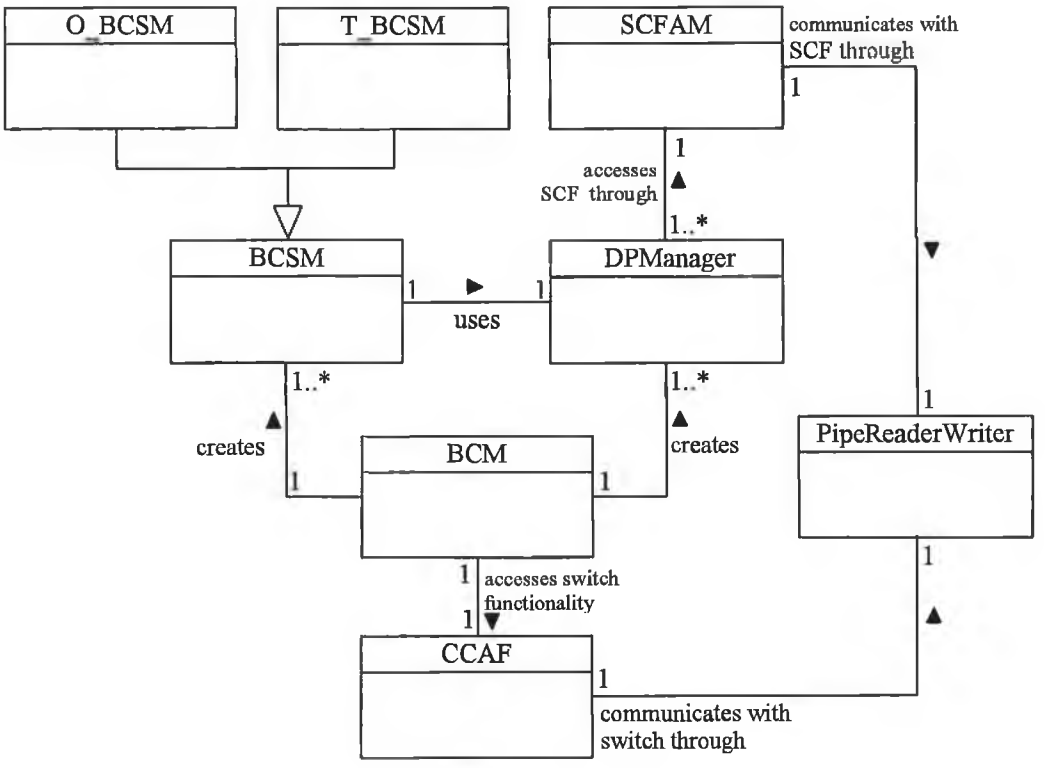


Figure 5.2 – Class Diagram for Module *ssf* of the IN Platform Prototype

Each of the PipeReaderWriter, CCAF, BCM and SCFAM classes are represented by one object (instance) in the *ssf* module. Instances of the O_BCSM and T_BCSM class are created as new calls come in and are destroyed when the calls end.

[17] For the testing of the prototype four telephones were used, configured to have numbers 1100, 1101,

Each O_BCSM or T_BCSM instance has an instance of the DPManager class associated with it, which is created and destroyed at the same time as the O_BCSM/T_BCSM. Detailed UML descriptions of the classes appearing in this high-level class diagram are given in Appendix F.

From the above it can be seen that the functionality of the CCAF, CCF and SSF Functional Entities are taken on by the described objects as follows: the CCAF FE functionality is contained in the CCAF object; the CCF FE functionality is divided between the CCAF, BCM and BCSM objects, while the SSF FE functionality is performed jointly by the BCSM, DPManager and SCFAM objects.

5.3.1.2.2 scf Module Object Model

The class diagram for the *scf* module is shown in Figure 5.3. The FEAM object allows the SCF access to all the other Functional Entities of the IN, in this case only the SSF. The FRL object contains the functional routines (implementations of SIBs) that are used in IN services. It also contains other data pertaining to IN services, as well as references to Service Logic Programs, which directly implement those services. The SLEM is the central object to the SCF FE functionality. It receives service requests, spawns SLPs, and keeps track of resources. The SDF object contains the entire SDF FE functionality. Detailed UML descriptions of the classes appearing in Figure 5.3 can be found in Appendix F.

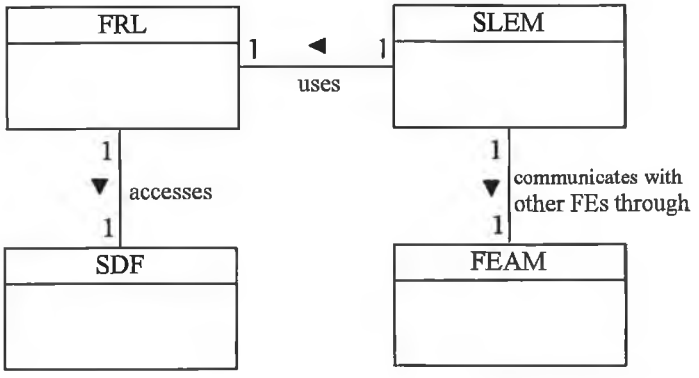


Figure 5.3 – Class Diagram for Module *scf* of the IN Platform Prototype

Each of the described classes are represented in the *scf* module with one object (instance).

1102 and 1103.

5.3.1.3 SIBs and Service Implementation

Service Independent Building Blocks (SIBs), as service-independent pieces of functionality, are part of the IN platform and a number of them have been implemented in the prototype to facilitate the provision of the sample IN services, which are described in the following section. The SIBs that have been implemented are: SERVICE DATA MANAGEMENT, USER INTERACTION and VERIFY. The SIBs have been iplemented as public operations of the FRL class. The Service Logic Programs (SLPs), chaining the SIBs and thus implementing the services, have been implemented as global functions linked into the *scf* module. The *scf* starts an SLP by running the SLP function on a separate thread.

How these SIBs have been chained to implement the sample services and inter-module communication for the services is shown in Appendix E.

5.3.2 Sample Services

5.3.2.1 Abbreviated Dialling

5.3.2.1.1 Service Description

This service allows the user to assign a telephone number to each digit from 0 to 9 and to place calls to those numbers by dialling a short sequence including the corresponding digit. The user assigns numbers to digits by dialling the <abdial_activate_digit_sequence> digit sequence, which can be described in BNF notation as follows:

<abdial_activate_digit_sequence> ::= #<abdialling_code>[<list_of_assignments>]##
<abdialling_code> ::= 14
<list_of_assignments> ::= <assignment>{<assignment>}
<assignment> ::= <identifying_digit><number>
<identifying_digit> ::= <digit>
<number> ::= <digit>{<digit>}
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

If <list_of_assignments> does not contain any assignments (that is, if #14## is dialled), all the assignments are cancelled.

The Abbreviated Dialling is performed by dialling the <abdialling_digit_sequence> digit sequence, which is as follows in BNF:

<abdialling_digit_sequence> ::= *<identifying_digit>
<identifying_digit> ::= digit
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

5.3.2.1.2 Demonstration Scenarios

- Abbreviated Dialling number assignment, use and de-assignment
  - ✓ 1100 dials #1411101#21102#31103##; 1100 receives positive confirmation tone
  - ✓ 1100 dials *1; 1101 rings, answers and is connected to 1100
  - ✓ 1100 dials *2; 1102 rings…
  - ✓ 1100 dials #141##; 1100 receives positive confirmation tone
  - ✓ 1100 dials *1; 1100 receives error tone
  - ✓ 1100 dials *3; 1103 rings…
  - ✓ 1100 dials #14##; 1100 receives positive confirmation tone
  - ✓ 1100 dials *3; 1100 receives error tone

5.3.2.2 Call Forwarding

5.3.2.2.1 Service Description

This service allows the user to forward calls, made to his/her telephone number, to another telephone number in the IN. The user activates/deactivates the Call Forwarding by dialling the <callforward_activate_digit_sequence> digit sequence, defined in BNF as follows:

<callforward_activate_digit_sequence> ::= #<callforward_code>[<target_number>]#
<callforward_code> ::= 36
<target_number> ::= <digit>{<digit>}
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

If the activation is done with a new target number, the target number is changed. If <target_number> is not present (i.e. #36# has been dialled), the Call Forwarding is deactivated.

5.3.2.2.2 Demonstration Scenarios

- Call Forwarding activation and deactivation, with their manifestations
  - ✓ 1100 dials #361101#; 1100 receives positive confirmation tone
  - ✓ 1102 dials 1100; 1101 rings, answers and is connected to 1102
  - ✓ 1100 dials #36#; 1100 receives positive confirmation tone
  - ✓ 1103 dials 1100; 1100 rings, answers and is connected to 1103

- chained Call Forwarding
  - ✓ 1100 dials #361101#; 1100 receives positive confirmation tone
  - ✓ 1101 dials #361102#; 1101 receives positive confirmation tone
  - ✓ 1103 dials 1100; 1102 rings, answers and is connected to 1103

5.4 The IIIN

5.4.1 Platform Prototype

5.4.1.1 Implementation Model

This platform prototype was built as an extension to the basic IN platform prototype. The IIIN implementation model is shown in Figure 5.4. The *comms* and *ssf* modules remain the same as in the basic IN platform prototype. The *scf* module has been modified to accommodate the IIIN functionality. The other modules of the IIIN platform prototype are the *iscgf*, which implements the Internet-Service Control Gateway Function, and the *iμs*, which implements the IIIN Micro Server, a part of the IUF (the other part being a WWW browser). The SSOISF and SPOISF consist of the office WWW server and a group of service-specific CGI programs.

While *comms*, *ssf* and *scf* are Windows NT programs, the *iscg* is a Unix executable, running on a Solaris 2.6 machine. It has been written in C++ and compiled using the SUN WorkShop C++ compiler. The *iμs* runs on either Windows 95 or Windows NT and has been implemented through C++ and compiled with Borland C++, like the basic IN modules.

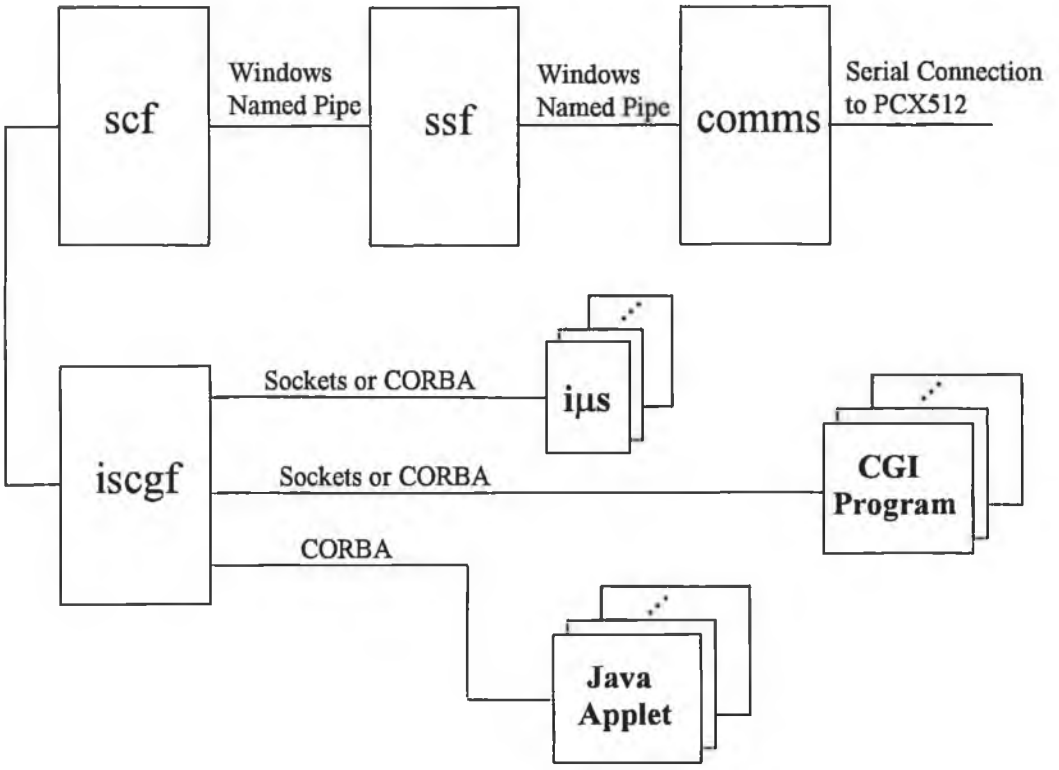


Figure 5.4 – Implementation Model for the IIIN Platform Prototype

The IIIN prototype contains one instance of the *iscgf* module and several instances of the *iμs* module, one for each end user of the IIIN. The *scf* and the *iscgf* communicate through TCP/IP sockets. The *iμs* modules communicate with other parts of the IIIN through the TCP/IP network i.e. the Internet, either using CORBA or sockets.

5.4.1.2 Object Models

The object model of the *scf* module in the basic IN part of the IIIN has not been changed from that for the basic IN. However, the FEAM, FRL and SLEM objects themselves have been changed. The FEAM has been extended to handle the communications between the SCF and the *iscgf* module. The FRL has been extended to include the IIIN SIBs and references to IIIN Service Logic Programs (SLPs). The SLEM has been changed so that it would process service requests from the Internet, apart from those from the SSF that it already handled in the basic IN prototype.

The class diagrams of the two modules comprising the IIIN extension to IN are very simple since they each contain one class, the ISCGF and IμS class,

respectively. Each of these classes is represented with one object (instance) in the appropriate module.

5.4.1.3 SIBs and Service Implementation

Four of the five IICSI1 SIBs, described in Chapter 4, have been implemented as part of the IIIN Prototype, in the *scf* module. Those are CALL INFORMATION UPDATE, RETREIVE CALL INFORMATION, SEND SERVICE COMPONENT REQUEST, and TIMER. The NON-REAL-TIME VOICE-CHANNEL DATA MANIPULATION SIB is not implemented since it is not needed for the implementation of the sample services. Of the two CS-1 SIBs used in the sample services, SDM had been implemented as part of the basic IN, while the UI SIB was implemented in this stage. The only CS-2-specific SIB, needed for the sample services, the END SIB has also been implemented. One Service-Dependent Building Block (SDB), CALL LIST, has been implemented, as it is part of the Group Announcements service, which is implemented here as a sample service. All the SIBs and the SDB mentioned above have been implemented as classes, defined within the IIIN, CS_1 and CS_2 classes, which, in their turn, are defined within the FRL class (see Appendix G).

Each sample IIIN service of the prototype is provided by the IIIN through the co-operation of the corresponding SLP in the SCF and other functionality, provided by entities in the Internet. Some of this other functionality is service-independent, such as the IIIN Micro Server (IμS) functionality and the Internet Service Control Gateway Function (ISCGF), while some of it is service-specific, such as the functionality contained in Java applets and CGI scripts. Descriptions of the IIIN service-specific pieces of functionality based in the Internet are contained in the following section, as part of each service description.

Appendix C contains figures of the actual SIB chains used to implement the Service Logic Programs (SLPs) corresponding to the sample IIIN services of the prototype. This is with the exception of the IIIN User Registration service feature, which is an example of Customer Profile Management, and for which the implementation SIB chains are given in Appendix E, since they differ from the general SIB chains for CPM.

Appendix G contains the C++ code listing for the two SLPs that implement the IIIN User Registration service feature.

5.4.2 Sample Services

5.4.2.1 Customer Profile Management (CPM)

5.4.2.1.1 Service Description

This service allows the user to manage customer-specific data pertaining to various services. In the instance implemented as a sample service for the IIIN prototype, the user registers for IIIN services in general, by specifying his/her telephone number, type of IP address and IP address, if it is statically allocated. The user registers to use the IIIN by filling in a form on an HTML page with the required data and submitting the form. Soon after, the user is presented with a WWW page, requesting him/her to make a phone call to a specific number in the telephone network and enter a code (also given on this Web page, when prompted, for the verification of the user's location. The user should then do as instructed and should receive a Web page confirming successful registration if he/she has followed the instructions properly. If at any stage of the CPM instance an error occurs in the system or the user makes a mistake, an appropriate Web page is presented to him/her.

5.4.2.1.2 Service-Specific Functionality

The service-specific functionality in the Internet for this service is contained in an HTML form, two other HTML pages and a CGI program that processes that form.

- The HTML form, called "IIIN Service Registration Form", is presented in Figure 5.5.
- The CGI script, "iiin_user_reg.cgi" (the C code listing for which may be found in Appendix G),
  1) extracts the user's telephone number from the submitted form
  2) generates a random number with 5 digits
  3) sends an IIINTBP message to ISCGF with a request for telephone number confirmation, containing the submitted telephone number and the generated random number (this request is sent on to the SCF to activate the appropriate SLP)
  4) constructs and returns a Web page requesting from the user to perform the telephone number confirmation procedure, with the telephone number to be called and the generated random number to be entered when prompted.

The CGI program also generates web pages to:

- ➢ to warn the user if he/she has made a mistake and failed to register;
- ➢ inform the user if an error occurred in the system.

- The "IIIN User Registration Confirmation" page is displayed to the user through a request to the IμS once the registration is successfully completed
- The "IIIN User Registration Invalid Code" page is displayed to the user through a request to the IμS if the user enters an invalid code through the phone.

Here all the Web pages and the CGI script are conceptually (since the lab for the prototype has access to one server only) based on a Service Provider-Owned Internet Server (SPOIS). An IμS with socket-type communication has been used.

5.4.2.1.3 Demonstration Scenarios

The demonstration of this service feature is tightly coupled with other services i.e. the effects that it creates are visible only during the execution of other services. In fact, if the user does not register he/she will not have access to most of the IIIN services and trying to invoke those services will fail. If those services are successfully invoked after registration, that is a good sign that the registration was successful.

It should be noted that the first demonstration scenario includes the downloading and installation of the IμS, an activity which is not part of the CPM feature.

Following are the demonstration scenarios:

- entirely successful IIIN user registration
  - ✓ the user loads the "Welcome to the IIIN" Web page (shown in Figure 5.5)
  - ✓ the user downloads the IμS by clicking on the Download IIIN Micro Server link
  - ✓ and clicks on the form link to the IIIN user registration form
  - ✓ the "IIIN User Registration Form" HTML form, presented in Figure 5.6a, is loaded into the browser (an example of an IIIN User Proxy registration form is shown in Figure 5.6b, although it is not used in the sample services) and filled in as shown in the figure
  - ✓ the "Register" button is clicked
  - ✓ the browser displays a Web page with further instructions to the user, shown in Figure 5.7

- ✓ according to the instructions on the Web page (Figure 5.7), from the phone with the address 1100 #00 is dialled, then 305834 (there are no UI facilities in the prototype); the receiver is put down
- ✓ an acknowledgement Web page (shown in Figure 5.8) is presented to the user (if the browser is not already running, it is automatically started)

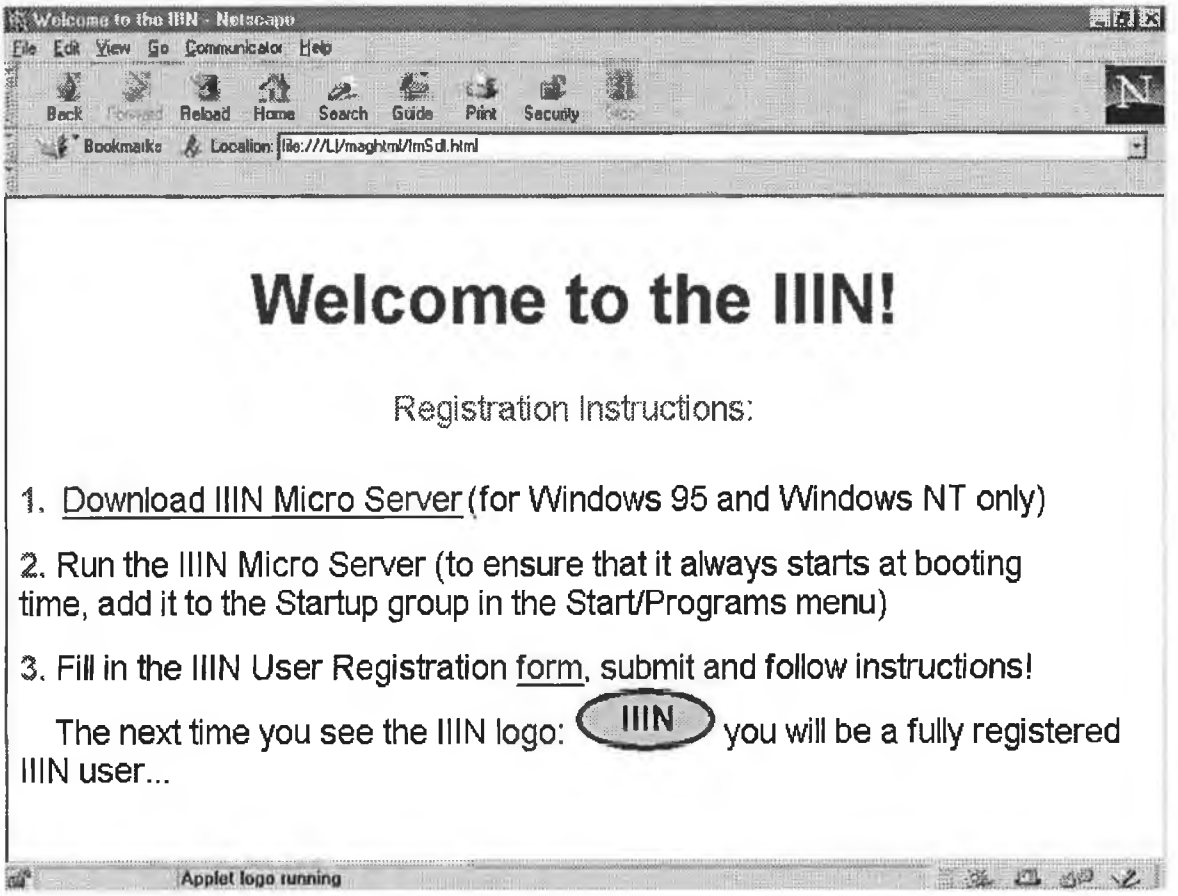
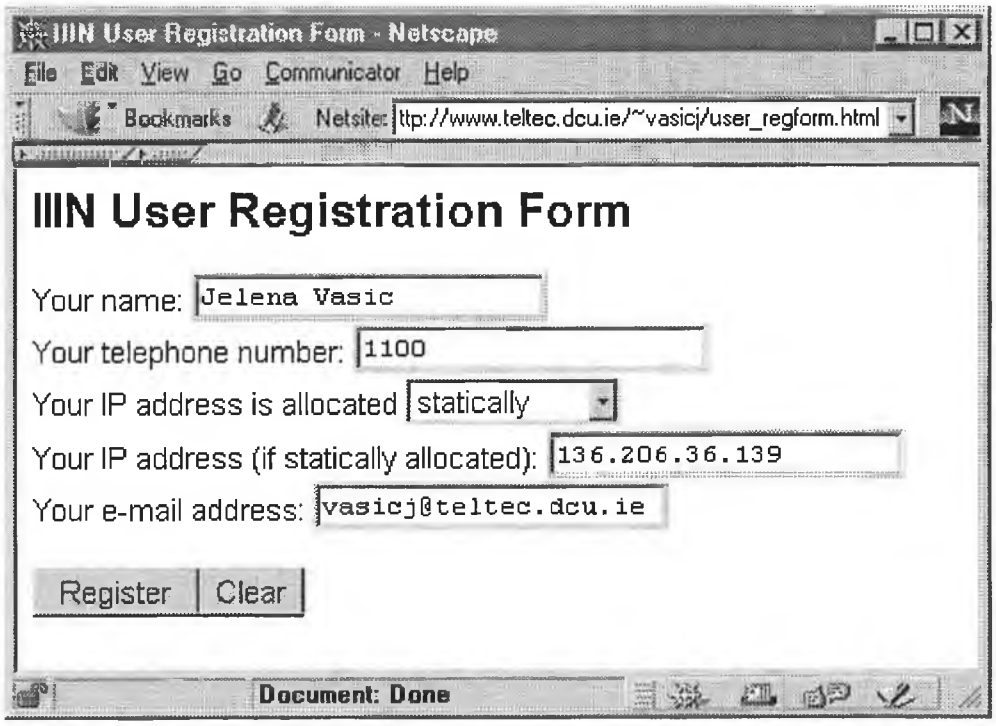
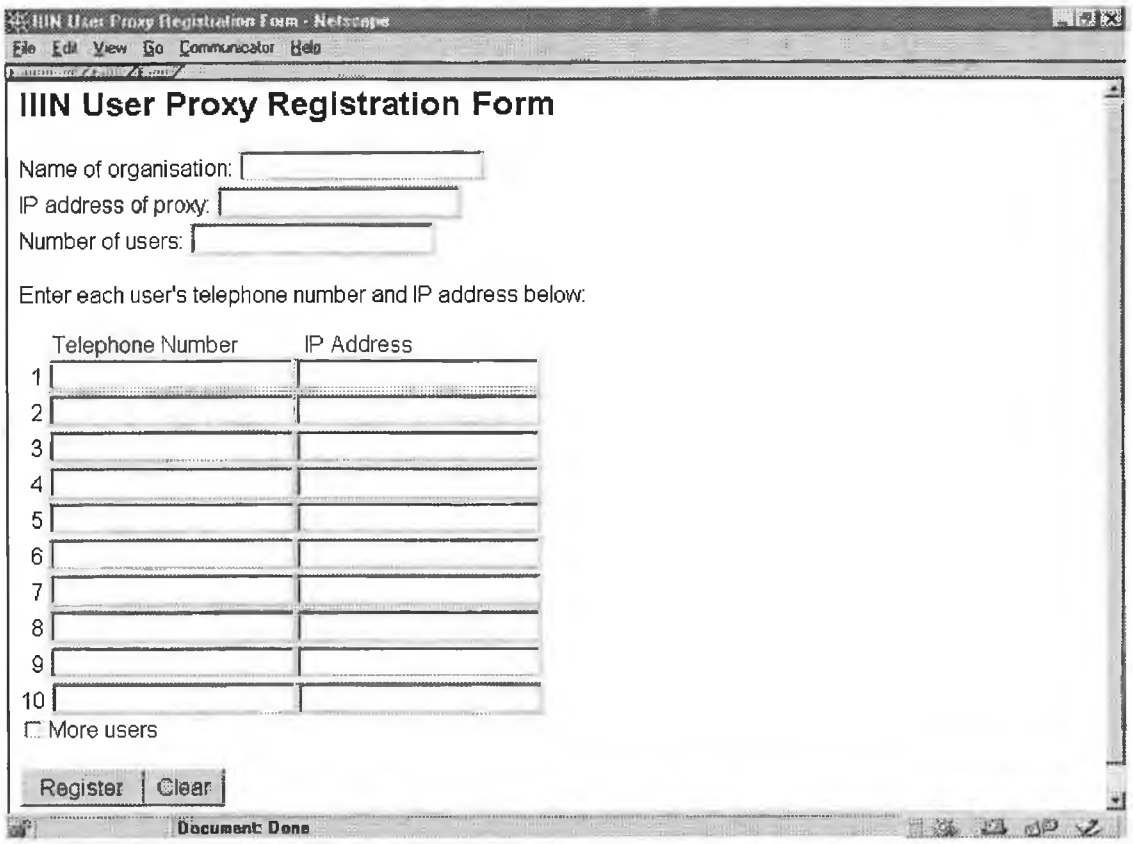


Figure 5.5 – WWW Page with Instructions for the Download of the IμS and for Subscribing to the IIIN, for a Single User

- unsuccessful IIIN user registration, caused by invalid form data
  - ✓ the "IIIN User Registration Form" HTML form, presented in Figure 5.6a, is loaded into the browser and filled in as in that Figure, except for the IP address which is given as 136.206.36
  - ✓ the "Register" button is clicked
  - ✓ the browser displays the error Web page shown in Figure 5.9



a) IIIN User Registration HTML Form



b) IIIN User Proxy Registration HTML Form

Figure 5.6 – IIIN User and User Proxy Registration Forms

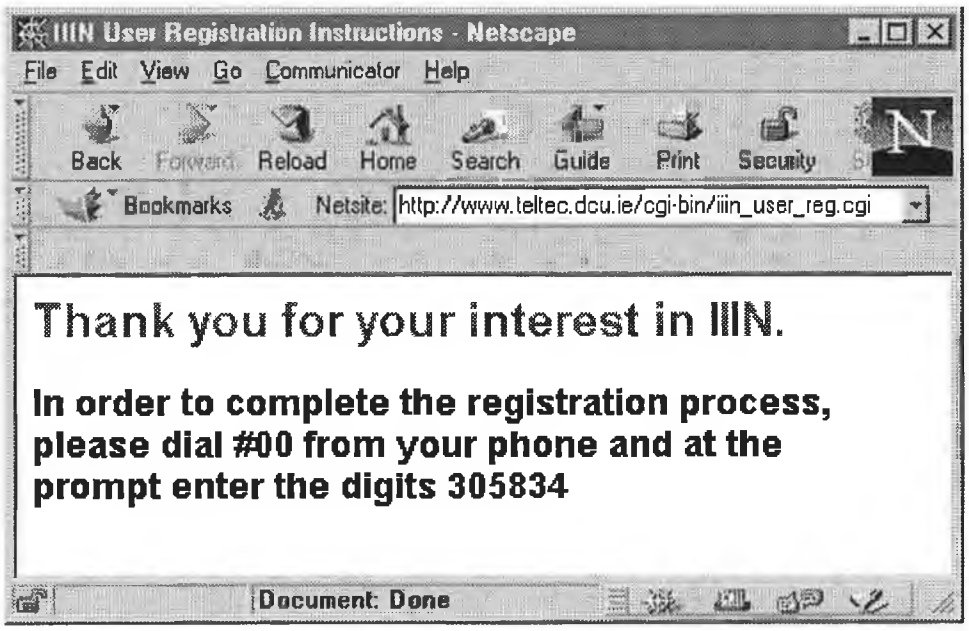


Figure 5.7 – IIIN User Registration Instruction Page, Produced by the IIIN Registration Service CGI Program

Figure 5.8 – IIIN Registration Service Confirmation Page

- unsuccessful IIIN user registration, caused by an invalid confirmation code entered from the telephone set
- the first 5 steps are the same as in the first scenario (entirely successful IIIN Service Registration)
  - ✓ from the phone with the address 1100 #00 is dialled, then 305843 (note that the code is incorrect); the receiver is put down
  - ✓ an error Web page (shown in Figure 5.10) is presented to the user (if the browser is not already running, it is automatically started)

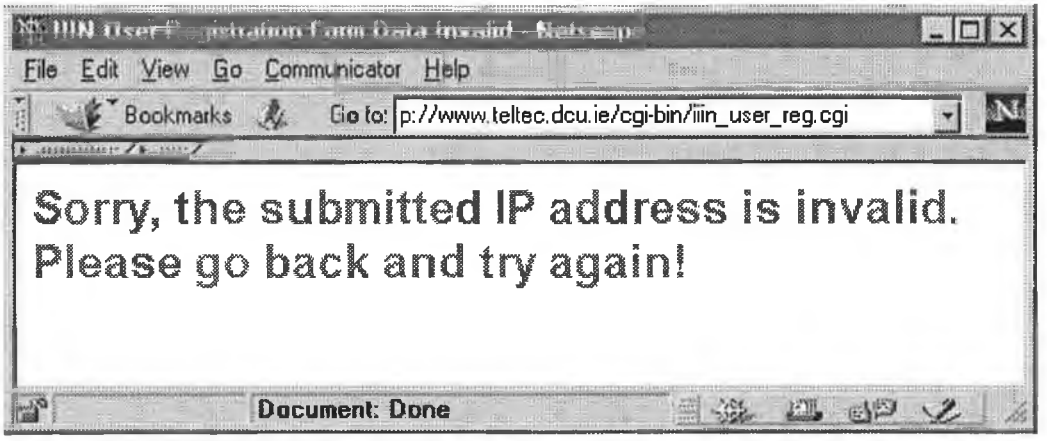


Figure 5.9 – IIIN Registration Form Data Invalid Error Page, Produced by the IIIN Registration Service CGI Program

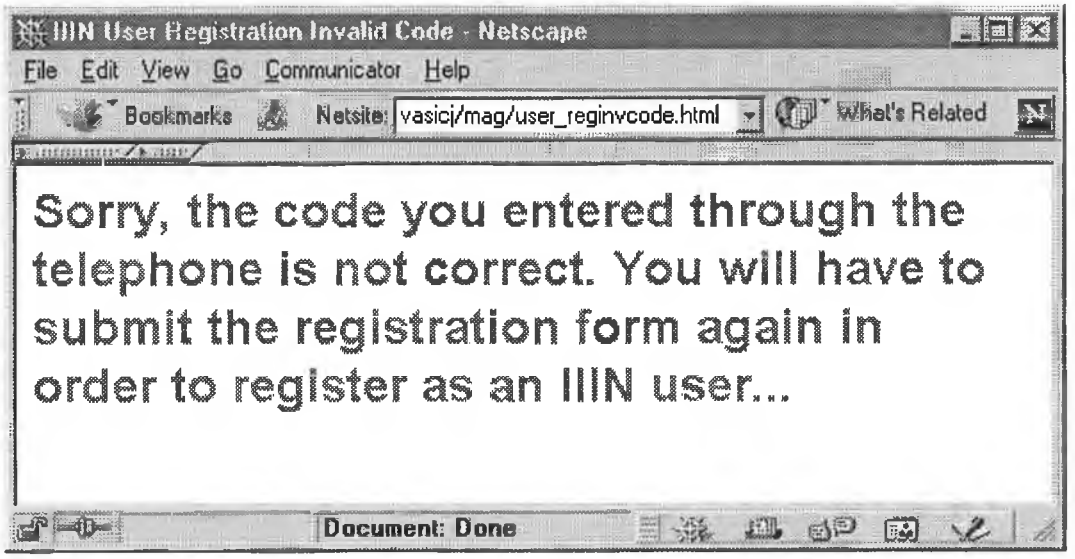


Figure 5.10 – IIIN Registration Invalid Code Error Page, Produced by the IIIN Registration Service CGI Program

5.4.2.2 Click-to-Dial-Back

5.4.2.2.1 Service Description

This service has many different variations of which only one has been implemented as a sample service. In this particular variation of the service the user who is browsing the WWW comes upon the Web page of some company it is interested in contacting. The Web page features a Java applet containing a "button" which the user is told to click on if he/she wishes to be contacted by an agent from the company. After clicking on the applet the user is given indication of a successful

request within the applet's graphics area and, immediately or after some time, he/she is called by an agent in the company to whom the Web page belongs.

5.4.2.2.2 Service-Specific Functionality

The only service-specific component of this service is the applet on the Web page provided by the service subscriber. The Web page with the CTDB applet, which has been used in the prototype, is shown in Figure 5.11. The CTDB applet uses a CORBA method invocation on the ISCGF to invoke the service. The ISCGF then conveys this request to the SCF, where the appropriate SLP is activated to first make a call to an agent, then connect her/him to the user who requested the call. The Web page containing the CTDB applet is (conceptually) based on a SSOISF. The Java code listing for the CTDB applet can be found in Appendix G.

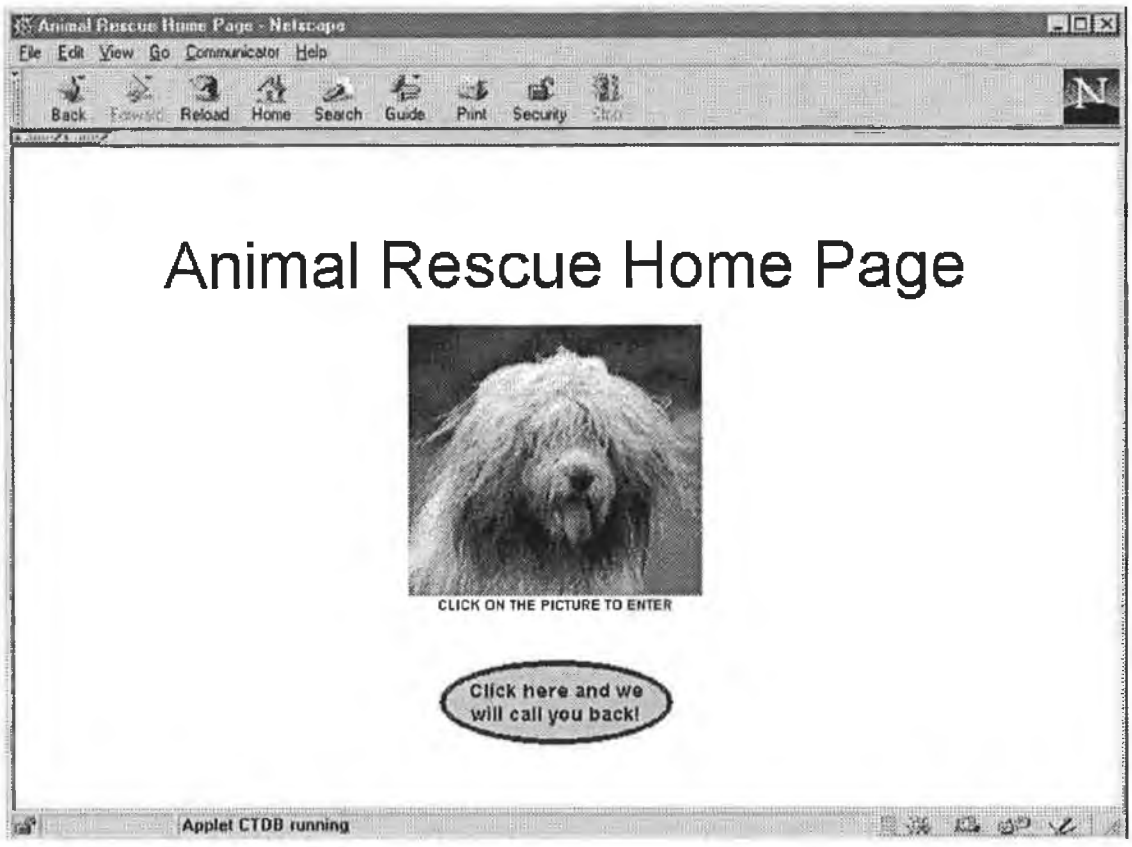


Figure 5.11 – Web Page with Click-to-Dial-Back Request Applet

5.4.2.2.3 Demonstration Scenarios

It should be noted that in order for this service to work, the user has to have registered, using the IIIN Registration Service, a type of the Customer Profile Management (CPM) feature. It is assumed here that the first scenario described above

for that service (entirely successful IIIN user registration) has taken place before the following scenarios.

- successful instance of the Click-to-Dial-Back service
  - ✓ the WWW page shown in Figure 5.11 is loaded into the user's browser and the applet saying "Click here and we will call you back!" is clicked on
  - ✓ the telephone with the number 1102 (associated with the page in Figure 5.11) rings and is answered
  - ✓ then the registered user's telephone (number 1100) rings and is answered
  - ✓ the call between 1102 and 1100 is connected

5.4.2.3 Group Announcements

5.4.2.3.1 Service Description

In order to place a group announcement, a user must access a Web page specially set-up for that purpose by the service provider on the Service-Provider-Provided Server (for the prototype, the only server in the lab). This Web page contains a form that is to be filled in, requesting various data. After successfully submitting the form, the user receives a page requesting him/her to make a call to a certain number, before a time-out (which should be substantial, for example 20 minutes) and at the prompt to say the message he/she would like to broadcast to the list of numbers. After a spoken or tone confirmation that the message is recorded, the user has completed his part of the job. Now the IN automatically makes calls to the numbers on the list and on answer plays the announcement that the user recorded. If the form is not properly filled in, the user receives a Web page declaring an error. If the user fails to call the given number and have a message recorded within the time specified, even if the form has previously been submitted with success, the service instance is aborted.

5.4.2.3.2 Service-Specific Functionality

The service-specific functionality for this service is similar to the components of the CPM service, described above. There is a Web page with a form, "Group Announcement Request Form", through which the user places a request for the service. This page for the sample Group Announcements service is shown in Figure 5.7. The other piece of service-specific functionality for this service is the a CGI script, "iiin_ga.cgi", which processes the submitted form and sends an IIINTBP

message to the ISCGF to request the group announcement. The ISCGF sends a message to the SCF, which activates the appropriate SLP to carry out the announcements.

5.4.2.3.3 Demonstration Scenarios

It should be noted that in order for this service to work, the user has to have registered, using the IIIN Registration Service, a type of the Customer Profile Management (CPM) feature. It is assumed here that the first scenario described above for that service (entirely successful IIIN user registration) has taken place before the following scenarios.

- successful instance of the Group Announcements service
  - ✓ at 9:45 (june 10th), the HTML form shown in Figure 5.12 is loaded into the user's browser and is filled in as in that Figure (this is done from the PC associated with phone with number 1100); the "Submit" button is clicked on

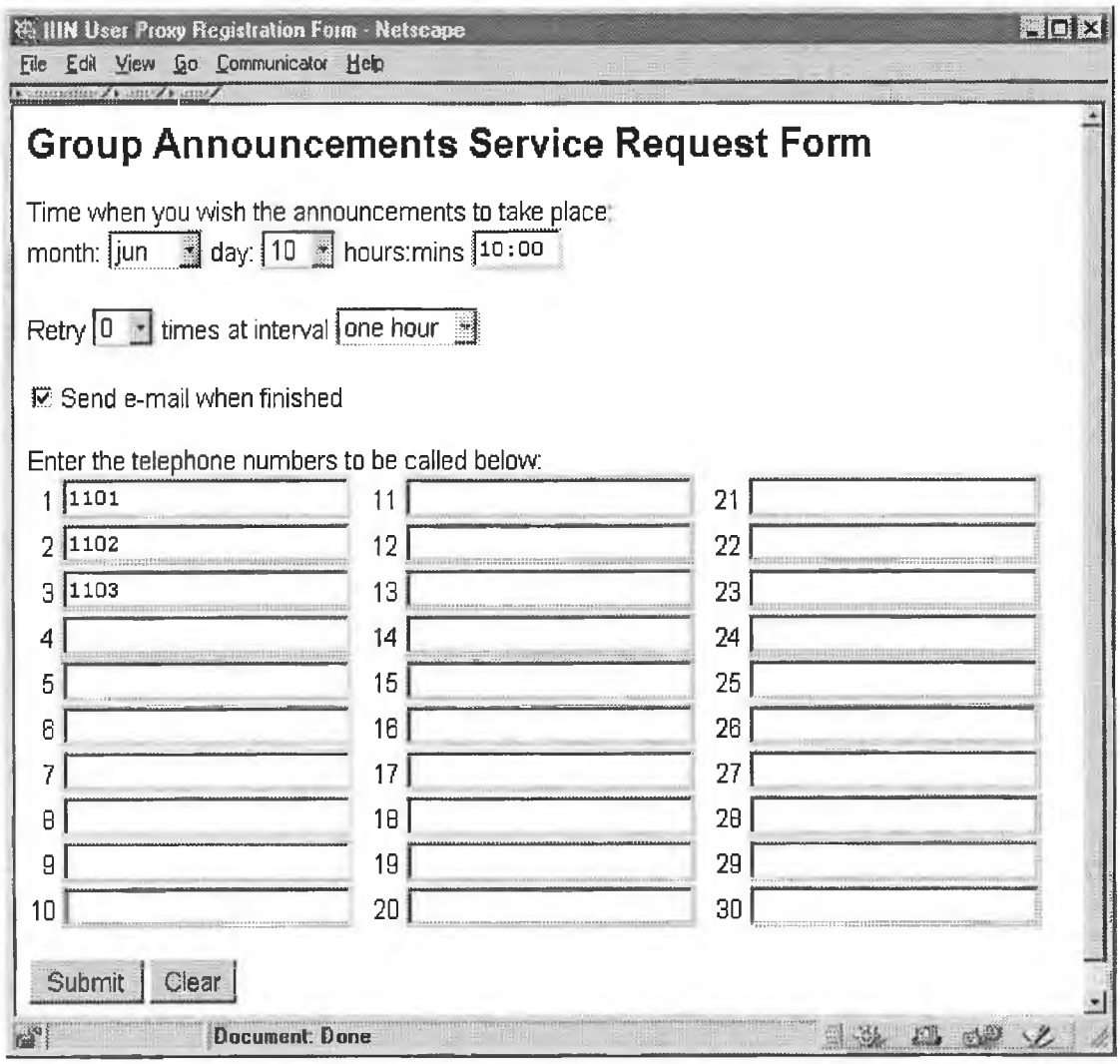


Figure 5.12 – Group Announcements Service Request HTML Form

- ✓ the Web page in Figure 5.13, constructed by the invoked CGI program, appears in the browser
- ✓ at 9:50 (june 10th), from the user's phone (1100), #01 is dialled and an acknowledgement tone is heard (no message is recorded since those capabilities are not part of the prototype); the receiver is put down
- ✓ at 10:00 (june 10th) the telephone with address 1101 rings and is answered, there is an acknowledgement tone (no announcement because of the limitations of the prototype), the receiver is put down; the same is repeated at telephone sets with numbers 1102 and 1103
- ✓ the user (with telephone number 1100 receives e-mail with confirmation of successfully completed service instance

- unsuccessful instance of the Group Announcements IIIN service, caused by the user not recording the message within 10 minutes of being instructed
  - ✓ at 10:10 (June 10th), the HTML form shown in Figure 5.12 is loaded into the user's browser and is filled in as in that Figure, except that the requested time for the announcements is set to 10:30 instead of 10:00 (this is done from the PC associated with phone with number 1100); the "Submit" button is clicked on
  - ✓ the Web page in Figure 5.13, constructed by the invoked CGI program, appears in the browser

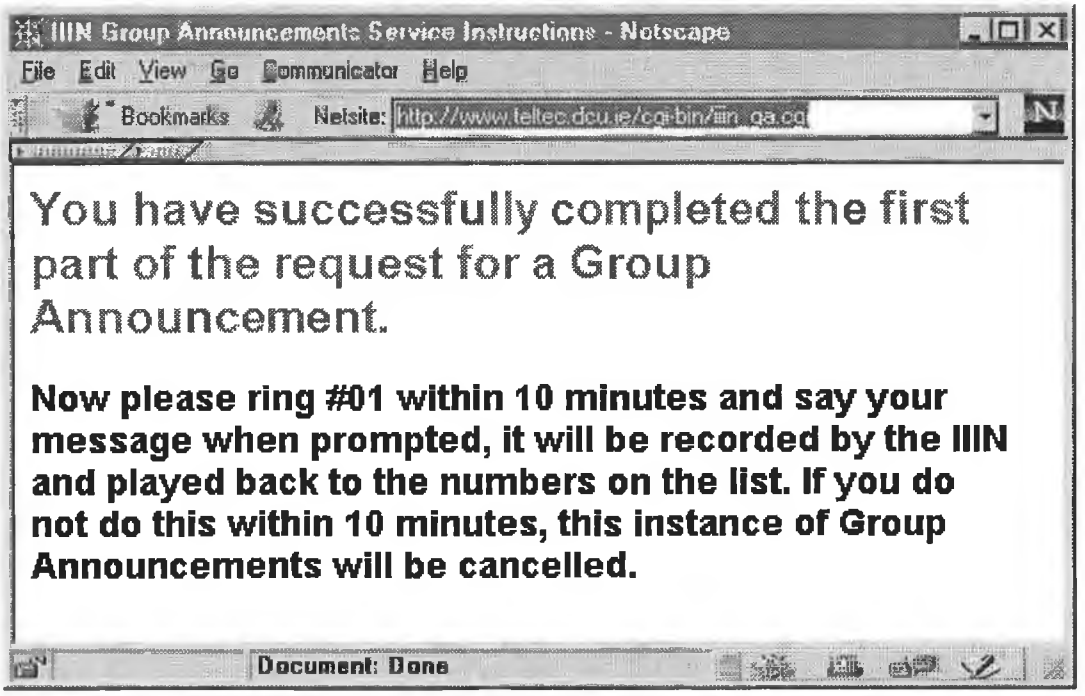


Figure 5.13 – Group Announcements Service Instructions Page, Constructed by the Group Announcements Service CGI Program

- ✓ at 10:25 (june 10th), from the user's phone (1100), #01 is dialled and an error tone is heard (no message is played since that capability is not part of the prototype), indicating that the service instance had been cancelled (it should have happened at 10:20, 10 minutes after the form submission)

5.4.2.4 Parallel Routing

5.4.2.4.1 Service Description

In the case of Parallel Routing, the service is activated by an agent at an IN service subscriber's site. The agent is talking to a customer on the phone. The agent decides to send some information (e.g. a picture of a house if the subscriber is an estate agent business) - in real time - to the user. The agent has the picture open in her browser and she clicks on a "button" which is a part of an applet present beside the picture on the Web page. Soon after the client on the other side of the line has the picture in front of him, open in his browser. The applet acknowledges the sending of the picture, or warns the agent of an error, if one occurs.

5.4.2.4.2 Service-Specific Functionality

In a general case, this service should include a complex database of objects that can be fetched and incorporated into HTML documents, stored in the service subscriber's server (the SSOISF). This database can be browsed by the subscriber's agents, using dynamic HTML documents. In the case of the IIIN prototype sample service, the agent has access to a Web page containing four pictures and beside each picture the PRSB (from Parallel Routing Send Button) applet. This Web page, "Animal Rescue Adoption Service", is shown in Figure 5.14. The PRSB applet is visible in the form of a "button" with the text "Send photo to client". When this button is clicked on, the applet makes a CORBA request, containing a reference to the graphical object (photo) being sent, to the ISCGF. The ISCGF activates an SLP in the SCF. This SLP detects the phone number of the client talking to the agent on the telephone (this dynamic information is retained in the SCF for all IIIN users in the prototype, although in a real-life implementation, where resources are scarce, it should be done only for organisations/companies specially subscribed to this service). The SLP then reads the matching IP address from its database (the client also has to be registered as an IIIN user) and launches a request, containing the same reference to

a graphical object that it has received from the PRSB applet, to the client's IμS. The client's IμS activates the client's browser to display the referred-to object.

For this service, an IμS with socket communication has been used.

5.4.2.4.3 Demonstration Scenarios

It should be noted that in order for this service to work, both the agent and client have to have registered, using the IIIN Registration Service. It is assumed here that the first scenario described above for the IIIN Registration Service (entirely successful IIIN user registration) has taken place for the two parties before the following scenario.

- successful instance of the Parallel Routing service
  - ✓ the client (telephone number 1102) rings the agent (telephone number 1100) and after the agent answers, they are connected
  - ✓ the WWW page shown in Figure 5.14 is accessed through the browser on the agent's PC
  - ✓ the agent clicks on one of the buttons with the text "Send photo to client"
  - ✓ the picture beside the button that the agent has clicked on appears in the client's browser (if the browser is not already running, it is started)

Figure 5.14 – Parallel Routing Data Page with Send Request Java Applets

5.5 Conclusion

This chapter has presented the implemented system, consisting of a basic IN prototype, extensions made to that prototype in order to create a prototype of the IIIN and sample services, both for the basic IN and for the IIIN.

While the comms module was written by another student, as well as parts of the ssf module, the remainder of the system has been implemented by the author herself. At the time of writing, most of the system has been implemented in full. The comms and scf modules, as well as all the Internet-based parts of the IIIN are complete and tested. So is the ssf module for the basic IN. The extension of the ssf module for the IIIN has been written but not fully tested, due to a technical problem with the PCX switch, which arose in the later stages of the project. Therefore, while the basic IN sample services have been tested on the full system, the IIIN sample services have only been tested with an ssf simulator (i.e. a program accepting signals from the computer console rather than the switch).

The platform and the services have been tested only functionally, in accordance with the aim of the project. The platform has the expected behaviour. It processes "ordinary" calls, detection points and invokes Service Logic when needed. The sample services manifest the desired functionality well in error-free circumstances, but have not been perfected to handle every exceptional situation. For example, a processing error in the SLP does not always result in the freeing of the suspended call. A better use of timers in both the ssf and the scf should solve this problem, without requirements for a large amount of additional code.

6 Conclusion

This project has examined the concept of integrating the Internet and the Intelligent Network (IN), concentrating on the special case of integration where both networks are involved in the resulting system in such a way as to partly or wholly provide the real-time components of the services offered by that system. End users participate in instances of such services through one or both networks simultaneously, as end users of both the IN and the Internet. Also, non-human operated end devices specific to the Internet, as well as those specific to the telephone network, are used for the provision of such services. The project has not looked at integration for the purposes of management and administration of the IN through the Internet or at the possible utilisation of the Internet to implement certain elements of the IN (e.g. the SCF and the SDF). It concentrates on the real-time communication aspect of the Internet rather than on its uses in distributed processing, distributed data storage and remote management and configuration.

Two types of gateway for the exchange of real-time information between the IN and the Internet have been identified: a gateway at the voice-channel level and a gateway at the control level. These two types of gateway produce two fundamentally different types of integrated systems, each with potential for providing a specific set of services. The gateway on the voice-channel level basically has the role of voice-channel information (voice or facsimile) format converter – between the native format of the telephone network and some appropriate format native to the Internet. The control-level gateway between the IN and the Internet allows for flow of control information (pertaining to calls in the telephone network as well as to activities in the Internet) between the two networks.

The approach taken in designing the integrated system was aimed at minimising the interactions between the two networks and keeping their distinct ways of functioning intact, while enabling a variety of services. Therefore, in the Internet Integrated Intelligent Network, the service control gateway allows only highly non-intrusive communication from the Internet, without any direct call-control granted through it. The IN carries out the functionality of "controlling" the special resource gateway transparently in relation to its normal functioning, by performing number

translations and directing the special resource in its activities. There are no IN-specific functional entities in the Internet. All the functionality needed for the IIIN is implemented through mechanisms already available – CGI scripts, HTML pages and Java applets.

However simple, this approach has shown to provide a platform for a large group of services, some of them based on known CT services and some specific to Internet-IN integration. A much researched case of the media gateway, Voice over IP integration with telecommunication networks, has not been studied here, but other services using that type of gateway, such as public Fax-on-Demand[18] and public Unified Messaging[19], have been described as services provided by the designed system. The control exercised through the control-level gateway can be in either direction, with actions in the Internet being controlled from the IN or actions in the IN being controlled from the Internet. Examples of services using the control-level gateway with control exercised by the Internet over the IN are Click-to-Dial[20] and Group Announcements[21]. An example of a service using the control-level gateway for control of activities in the Internet by entities the IN appears in the IN-implemented version of Automatic Call Distribution, which includes the screen pop-up feature[22]. Finally, the control taking place can be passive rather than active as in the previous examples. A service-providing entity in the Internet may request static or dynamic information, needed for completing the service instance, from the IN. An example of a service where dynamic (service instance-related) information is requested from the IN is Parallel Routing[23], while an example where static (general service/user related) information is requested from the IN is Email-by-Number.

Since many of these services, both those using the voice-channel gateway and those using the control gateway, have been modelled after classical examples of Computer Telephony Integration third-party-control-driven applications, some of them could also be implemented using a PBX connected to the Internet. However, in the area of work-from-home arrangements and geographically distributed businesses,

---

[18] Allows a request to be placed from the telephone network, which results in the faxing of a document stored in the Internet to a number in the telephone network.
[19] Allows voice messages, email and facsimile for the same user to be received in the same "mailbox" on the Internet.
[20] Allows a user to request, through the Internet, for a call to be placed in the telephone network.
[21] Allows a subscriber to request, through the Internet, that an announcement be made to a list of numbers in the telephone network.
[22] Pops up information about a customer to the screen of an agent, while the agent is receiving a call from the customer.
[23] Allows automatic routing of data between two users on the Internet, while they are conducting a telephone conversation.

as well as for businesses already subscribing to the Virtual Private Network (VPN) IN service, the IN solution is the only possible one. Also, certain services can be offered to the individual users of the IN and Internet.

As has already been mentioned, the services described in this project have been limited to those that do not involve any call-control originating in the Internet, once a call in the telephone network has been initiated. This excludes the possibility of providing services such as call deflection and multiparty call visual manipulation. Such services have been left out with the aim of limiting the scope of the project but also because of the current real-time-performance limitations and security issues associated with the Internet, which would affect this kind of service more than others.

In comparison with existing systems that integrate the telephone network and the Internet, and especially those described in Chapter 3, the IIIN is similar in structure with the more tightly integrated systems, such as the Lucent system (§3.4.3.2), in that the telephone network is linked to the Internet directly through the SCF (or its equivalent). The presence of the IμS in the IIIN has the same role as the use of the Finger protocol in the Siemens Web Call Centre (§3.4.3.2). Therefore, systems similar to the IIIN have already been built. However, presently they exist only in the form of proprietary products. For services of this type to be deployed on the all-encompassing scale of the telephone network or the Internet, as they are each separately, there would have to be a standard and universal integrating functionality present in the network(s). This project has had the aim to investigate what requirements should be imposed on the IN, in order for it to represent the base for a system such as the IIIN.

In the project, an accent was placed on the identification of additional SIBs (apart from those defined in CS-1 and CS-2) that would be needed for the provision of services specific to an Internet-IN integrated system. First of all, most of the IIIN services have components initiated from both the switching entities and the Internet, which are independently created but nevertheless need to exchange information and to be synchronised. In other words, there is a need for communication between Service Logic Programs that are not in a parent-child relationship and which therefore cannot make use of the Message Handler (MH) CS-2 SIB. This functionality has been assigned to the Call Information Update (CIU)/Retrieve Call Information (RCI) pair of SIBs. Another service independent piece of functionality needed for the IIIN is the initiation of requests towards entities in the Internet and this role is embodied in the Send Service Component Request (SSCR) SIB. Further, control needs to be exerted

over the Special Resource Gateway and the entities attached to it on the IN and Internet side. This is done by the Non-Real-Time Voice-Channel-Data Manipulation (NRT-VCDM) SIB. Finally, a TIMER SIB has been introduced to cater for some delayed activities that are part of IIIN services.

The process of identifying the necessary IIIN SIBs helped in the isolation of a group of particular requirements imposed on an IN that would support IIIN functionality, each requirement met in a separate SIB. With the use of SIBs in the implementation of the sample services, the functionality that they provide was reused in a way extremely convenient for the programmer, even though in some cases the use of a SIB for the sake of formality may lead to the introduction of unnecessary code and the slowing down of real systems with high performance requirements.

As for possible further research, work done in this project may be expanded in the future in several different ways. First of all, a study could be undertaken of the most pressing issues from the point of view of possible implementation and deployment of the IIIN - security and performance. The investigation of solutions to each of these problems could be material for a large project in itself. Next, if these problems were solved, there would be space for tighter integration (at the service control level) between the IN and the Internet, with the Internet gaining more control over the telephone network, and services such as visual manipulation of multiple-party telephone calls, real-time call deflection and call transfer, all done through the Internet. Further, an API could be developed, similar to those used in Computer Telephony Integration, for access to IN capabilities by programmers implementing IIIN services, or already existing CT APIs could be adapted for use with the IIIN. Another interesting topic for research would be the possible use of IN in performing the gateway function between Voice-over-IP and the traditional telephone network.

The IIIN platform prototype and the sample service prototypes that have been implemented demonstrate the concept of the above-described integration of the Internet with the IIIN. For the moment, the questions of reliability and security on the Internet demand a certain open-mindedness from potential service providers. On the other hand, the nature of the services chosen to be described in this project is such that their attractiveness and usefulness do not depend primarily on performance, but are based on the innovative and "lateral" additions that those services offer to the mechanisms of daily communication in the computerised world of today – without demanding expensive new equipment or software to be bought by the user.

References

[Ansc96] Thomas A. Anschutz, Lucent Technologies, "A Historical Perspective of CSTA", IEEE Communications Magazine, April 1996

[AVT97] Applied Voice Technology Inc., "Computer Telephony and the Internet (An AVT Perspective) White Paper", AVT, 1997,
@ ftp://ftp.telephoneintelligence.com/pub/WPrinter.pdf

[Benn93] Ronnie Lee Bennett and George E. Policello II, "Switching Systems in the 21st Century", IEEE Communications Magazine, March 1993

[Bern96] Tim Berners-Lee, "The World Wide Web: Past, Present and Future", August 1996, @ http://www.w3.org/People/Berners-Lee-Bio.html/1996/ppf.html

[Bern98] David Bernstein, "An IN and PSTN Signalling <-> Internet Protocols Interworking Unit and Programming Model for Network Based Transparent PSTN, VoIP & Web Integration", Proceedings of the 5th International Conference on Intelligence in Networks, May 1998

[Bout96] Thomas Boutell, "CGI Programming in C & Perl", Addison-Wesley Developers Press, 1996

[Burg98] F. Burghardt and W. Kirsch, Siemens AG, "Web Based Customer Service Control within the IN System of SIEMENS", Proceedings of the 7th IEEE Intelligent Network Workshop, Bordeaux, France, 1998

[Come97] Douglas E. Comer, "Computer Networks and Internets", Prentice-Hall, 1997

[Conr99] Lawrence Conroy, Scott Petrack, "The PINT Profile of SIP and SDP: a Protocol for IP Access to Telephone Call Services" (IETF Internet-Draft), IETF 1999, @ http://www.ietf.org/internet-drafts/draft-ietf-pint-profile-04.txt

[Cron96] Paul Cronin, Novell Inc., "An Introduction to TSAPI and Network Telephony", IEEE Communications Magazine, April 1996

[Dupu95] Fabrice Dupuy, Gunnar Nilsson and Yuji Inoue, "The TINA Consortium: Toward Networking Telecommunications Information Services", IEEE Communications Magazine, November 1995

[ECMA217] European Computer Manufacturers Association, "Services for Computer Supported Telecommunications Applications (CSTA) Phase II" (ECMA-217 Standard), December 1994

[ECMA269] European Computer Manufacturers Association, "Services for Computer Supported Telecommunications Applications (CSTA) Phase III" (ECMA-269), June 1998

[ECTF] ECTF Home Page @ http://www.ectf.org

[Fayn97] Igor Faynberg, Lawrence R. Gabuzda, Marc P. Kaplan, Nitin J. Shan, "Intelligent Network Standards (Their Application to Services)", McGraw-Hill, 1997

[Flan97] David Flanagan, "Java in a Nutshell", O'Reilly, 1997

[Fleg96] Robb Flegg, Mitel, "Computer Telephony Architectures: MVIP, H-MVIP and Scbus", IEEE Communications Magazine, April 1996

[Garr93] James J. Garrahan, Peter A. Russo, Kenichi Kitami and Roberto Kung, "Intelligent Network Overview", IEEE Communications Magazine, March 1993

[Gull98] Bjorn Gulla et. al., Telenor, "Improving IN Usability with GIN", Proceedings of the 5th International Conference on Intelligence in Networks, May 1998

[Hoog96] Herman D'Hooge, Intel Architecture Labs, "The Communicating PC", IEEE Communications Magazine, April 1996

[IEC99] The International Engineering Consortium, "Internet Telephony", IEC Tutorial 1999, @ http://www.webproforum.com/int_tele/index.html

[Irvi98] G. Irvine and N. Raguideau, Hewlett Packard, "A Futuristic Service Architecture for Personal Interactive Communications", Proceedings of the 7th IEEE Intelligent Network Workshop, Bordeaux, France, 1998

[Low96] Colin Low, "The Internet Telephony Red Herring", IEEE GLOBECOM 1996 – GLOBAL INTERNET '96, CONFERENCE RECORD – COMMUNICATIONS: THE KEY TO GLOBAL PROSPERITY, 1996

[Lu98] Hui-Lan Lu, "A Functional Architecture for IN Support of the Internet – The ITU-T SG11 View", The 42nd IETF Meeting, Chicago 1998, @ http://www.bell-labs.com/mailing-lists/pint/itu-sg11/sld001.htm

[McCo97] Brian McConnell, "CTI Software Overview", Hello Direct, Inc., December 1997 @ http://www.phonezone.com/tutorial/cti-overview.htm

[Mine98] Roberto Minerva, Corrado Moiso, Gabriele Viviani, "The Middleware ate the Verge between Internet and Telecom Services", Proceedings of the 5th International Conference on Intelligence in Networks, May 1998

[OMG95] Object Management Group, "The Common Object Request Broker: Architecture and Specification, Revision 2.0", OMG, 1995,
@ http://www.infosys.tuwien.ac.at/Research/Corba/OMG/

[OMG98] Teltec DCU et. al., "Interworking Between CORBA and TC Systems" (Final RFP Submission: telecom/98-10-03), OMG, October 1998

[Oppl97] Rolf Oppliger, "Internet Security: Firewalls and Beyond", Communications of the ACM, May 1997

[Ovum93] Man-Sze Li and Eirwen Nichols, "Intelligent Networks: Strategies for Customised Global Services" (Report), Ovum Ltd., 1993

[PINT99] IETF Secretariat, "PSTN and Internet Interworking (pint) Charter", IETF 1999, @ http://www.ietf.org/html.charters/pint-charter.html

[Q1200] ITU-T, "General Series Intelligent Network Recommendation structure" (Recommendation Q.1200), ITU-T, 1999

[Q1201] CCITT, "Principles of Intelligent Network Architecture" (Recommendation Q.1201), CCITT, 1993

[Q1202] ITU-T, "Intelligent Network – Service Plane Architecture" (Recommendation Q.1202), ITU-T, 1998

[Q1203] ITU-T, "Intelligent Network – Global Functional Plane Architecture" (Recommendation Q.1203), ITU-T, 1998

[Q1204] ITU-T, "Intelligent Network Distributed Functional Plane Architecture" (Recommendation Q.1204), ITU-T, 1993

[Q1205] ITU-T, "Intelligent Network Physical Plane Architecture" (Recommendation Q.1205), ITU-T, 1993

[Q1208] ITU-T, "General Aspects of the Intelligent Network Application Protocol" (Recommendation Q.1208), ITU-T, 1998

[Q1210] ITU-T, "Q.1210-Series Intelligent Network Recommendation Structure" (Recommendation Q.1210), ITU-T, 1996

[Q1211] ITU-T, "Introduction to Intelligent Network Capability Set 1" (Recommendation Q.1211), ITU-T, 1993

[Q1213] ITU-T, "Global Functional Plane for Intelligent Network CS-1" (Recommendation Q.1213), ITU-T, 1996

[Q1214] ITU-T, "Distributed Functional Plane for Intelligent Network CS-1" (Recommendation Q.1214), ITU-T, 1996

[Q1215] ITU-T, "Physical Plane for Intelligent Network CS-1" (Recommendation Q.1215), ITU-T, 1996

[Q1218] ITU-T, "Interface Recommendation for Intelligent Network CS-1" (Recommendation Q.1218), ITU-T, 1996

[Q1219] ITU-T, "Intelligent Network User's Guide for Capability Set 1" (Recommendation Q.1219) ITU-T, 1994

[Q1221] ITU-T, "Introduction to Intelligent Network Capability Set 2" (Recommendation Q.1221) ITU-T, 1998

[Q1222] ITU-T, "Service Plane for Intelligent Network Capability Set 2" (Recommendation Q.1222), ITU-T, 1998

[Q1223] ITU-T, "Global Functional Plane for Intelligent Network Capability Set 2" (Recommendation Q.1223), ITU-T, 1998

[Q1224] ITU-T, "Distributed Functional Plane for Intelligent Network Capability Set 2" (Recommendation Q.1224), ITU-T, 1998

[Q1225] ITU-T, "Physical Plane for Intelligent Network Capability Set 2" (Recommendation Q.1225), ITU-T, 1998

[Q1228] ITU-T, "Interface Recommendation for Intelligent Network Capability Set 2" (Recommendation Q.1228), ITU-T, 1999

[Q1290] ITU-T, "Glossary of Terms Used in the Definition of Intelligent Networks" (Recommendation Q.1290), ITU-T, 1998

[Q771] ITU-T, "Signalling System No.7 – Functional Description of Transaction Capabilities" (Recommendation Q.771), ITU-T, 1994

[RFC821] Jonathan B. Postel, "Simple Mail Transfer Protocol" (IETF RFC 821), August 1982, @ http://www.ietf.org/rfc/rfc0821.txt

[RFC854] J. Postel, J. K. Reynolds, "Telnet Protocol Specification" (IETF RFC 854), August 1983, @ http://www.ietf.org/rfc/rfc0854.txt

[RFC959] J. Postel, J. K. Reynolds, "File Transfer Protocol" (IETF RFC 959), October 1985, @ http://www.ietf.org/rfc/rfc0959.txt

[RFC1738] T. Berners-Lee et. al., "Uniform Resource Locators (URL)" (IETF RFC 1738), December 1992 @ http://www.ietf.org/rfc/rfc1738.txt

[RFC1866] T. Berners-Lee, MIT/W3C, D.Connolly, "Hypertext Markup Language – 2.0" (IETF RFC 1866), November 1995, @ http://www.ietf.org/rfc/rfc1866.txt

[RFC1883] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification" (IETF RFC 1883), December 1995, @ http://www.ietf.org/rfc/rfc1883.txt

[RFC2068] R. Fielding et. al., "Hypertext Transfer Protocol – HTTP/1.1" (IETF RFC 2068), January 1997, @ http://www.ietf.org/rfc/rfc2068.txt

[RFC2458] H.Lu et. al., "Toward the PSTN/Internet Inter-Networking – Pre-PINT Implementations" (IETF RFC 2458), November 1998,

@ http://www.ietf.org/rfc/rfc2458.txt

[Salu95] Peter H. Salus, "From ARPANET to Internet and Beyond", Addison-Wesley Publishing Company Inc., 1995

[Stam94] David A. Stamper, "Local Area Networks", The Benjamin/Cummings Publishing Company Inc., 1994

[Stra96] Carl R. Strathmeyer, Dialogic Corporation, "An Introduction To Computer Telephony", IEEE Communications Magazine, May 1996

[Titt96] Ed Tittel, Dawn Rader, "Computer Telephony: Automating Home Offices and Small Business", Boston: AP Professional, 1996

[Udel94] Jon Udell, "Computer Telephony", BYTE, July 1994

[Webe98] Geraldine M. Weber, "Netowork Intelligence – A Redefinition of the Network and the Meaning of Network Intelligence", Proceedings of the 5th International Conference on Intelligence in Networks, May 1998

[Zorn98] A. Zornak, Siemens AG, "Symbiosis of IN, TINA and Internet", Proceedings of the 7th IEEE Intelligent Network Workshop, Bordeaux, France, 1998

Acronyms

3pISRGF*	Third Party Provided ISRGF
3pSRC*	Third Party Provided Special Resource Controller
3pSRD*	Third Party Provided Special Resource Device
ACD	Automatic Call Distribution
AD	Adjunct
API	Application Programming Interface
AVD	Alternating Voice and Data
BCP	Basic Call Process
BCSM	Basic Call State Model
BCUSM	Basic Call-Unrelated State Model
B-ISDN	Broadband ISDN
CCAF	Call Control Access Function
CCF	Call Control Function
CCIS	Common Channel Interoffice Signalling
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CF	Call Forwarding
CGI	Common Gateway Interface
CID	Call Instance Data
CIU*	Call Information Update
CLI	Calling Line Identification
CORBA	Common Object Request Broker Architecture
CPH	Call Party Handling
CPM	Customer Profile Management
CSTA	Computer Supported Telecommunication Applications
CS-x	Capability Set x
CT	Computer Telephony
CTI	Computer Telephony Integration
CUSF	Call Unrelated Service Function
CVS	Call View States
DARPA	Defense Advance Research Projects Agency
DFP	Distributed Functional Plane
DHCP	Dynamic Host Configuration Protocol
DP	Detection Point

DPE	Distributed Processing Environment
DSDC	Direct Service Dialling Capabilities
DSVD	Digital Simultaneous Voice and Data
DTMF	Dual Tone Multiple Frequency
ECMA	European Computer Manufacturers' Association
ECTF	Enterprise Computer Telephony Forum
EDP	Event Detection Point
EF	Elementary Function
ETSI	European Telecommunication Standards Institute
FE	Functional Entity
FEA	Functional Entity Action
FEAM	Functional Entity Access Manager
FTP	File Transfer Protocol
GFP	Global Functional Plane
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IμS*	IIIN Micro Server
IASC*	Internet Application Service Component
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IF	Information Flow
IICSI*	Internet Integration Capability Set
IIIN*	Internet Integrated Intelligent Network
IIINTBP*	IIIN Text-Based Protocol
IIOP	Internet Inter-ORB Protocol
IIP	Internet Intelligent Peripheral
IN	Intelligent Network
INAP	Intelligent Network Application Part
INCM	Intelligent Network Conceptual Model
INWATS	Inward Wide Area Telecommunications Service
IP	Internet Protocol / Intelligent Peripheral
ISCGF*	Internet – Service Control Gateway Function
ISCGP*	Internet – Service Control Gateway Point
ISDN	Integrated Services Digital Network
ISRGF*	Internet – Special Resource Gateway Function
ISRGP*	Internet – Special Resource Gateway Point
IT	Information Technology
ITU	International Telecommunication Union

ITU-T	Telecommunication Standardisation Sector of the ITU
IUF*	Internet User Function
IUPF*	Internet User Proxy Function
IUWS*	Internet User Workstation
IVR	Interactive Voice Response
JTAPI	Java Telephony API
LAN	Local Area Network
MAN	Metropolitan Area Network
MH	Message Handler
MIB	Management Information Base
N-ISDN	Narrow-band ISDN
NRT-VCDM*	Non-Real-Time Voice-Channel Data Manipulation
NSF	National Science Foundation
O_BCSM	Originating BCSM
OMG	Object Management Group
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PCS	Personal Communication Service
PE	Physical Entity
PIC	Point In Call
PIN	Personal Identification Number
PINT	PSTN and Internet Inter-working
POB*	Point Of Beginning
POE*	Point Of Extension
POI	Point Of Initiation
POR	Point Of Return
POTS	Plain Old Telephone Service
PP	Physical Plane
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
PTN	Personal Telecommunication Number
RBOC	Regional Bell Operating Company
RCI*	Retrieve Call Information
RFC	Request For Comments
ROSE	Remote Operations Service Element
SCEF	Service Creation Environment Function
SCEP	Service Creation Environment Point
SCF	Service Control Function

SCFAM	Service Control Function Access Manager
SCP	Service Control Point
SCTP	Simple Computer Telephony Protocol
SCUAF	Service Control User Access Function
SDF	Service Data Function
SDK	Software Development Kit
SDL	Specification and Description Language
SDM	Service Data Management
SDN	Software Defined Network
SDP	Service Data Point
SF	Service Feature
SIB	Service-Independent Building Block
SLEE	Service Logic Execution Environment
SLEM	Service Logic Execution Manager
SLP	Service Logic Program
SMAF	Service Management Access Function
SMAP	Service Management Access Point
SMF	Service Management Function
SMP	Service Management Point
SMS	Service Management System
SMTP	Simple Mail Transfer Protocol
SN	Service Node
SNMP	Simple Network Management Protocol
SP	Service Plane
SPC	Stored Program Control
SPOIS*	Service Provider Owned Internet Server
SPOISF*	Service Provider Owned Internet Server Function
SRAIF*	Special Resource Assisting Internet Function
SRAIP*	Special Resource Assisting Internet Point
SRF	Special Resource Function
SS7	Signalling System Number 7
SSCP	Service Switching and Control Point
SSCR*	Send Service Component Request
SSD	Service-Specific Data
SSF	Service Switching Function
SSOIS*	Service Subscriber Owned Internet Server
SSOISF*	Service Subscriber Owned Internet Server Function
SSP	Service Switching Point

SSTP	Service Support Transfer Protocol
SVD	Simultaneous Voice and Datas
T_BCSM	Terminating BCSM
TAPI	Telephony API
TCAP	Transaction Capabilities Application Part
TCP	Transmission Control Protocol
TDP	Trigger Detection Point
TINA	Telecommunications Information Networking Architecture
TMN	Telecommunication Management Network
TSAPI	Telephony Services API
UDP	User Datagram Protocol
UML	Unified Modelling Language
URL	Universal Resource Locator
VPN	Virtual Private Network
WAN	Wide Area Network
WWW	World Wide Web

NOTE: The acronyms marked with an asterisk (*) have been defined in this project.

Appendix A

Stage 1 Description of the IICSI1 SIBs

CALL INFORMATION UPDATE (CIU)

a) Definition

This SIB makes note of some dynamic status pertaining to a resource in the IN.

b) Operation

This SIB preserves some dynamic information received from the Basic Call Process in a POI and not otherwise obtainable within the service logic, regardless of whether it will be used in an instance of the relevant service. The dynamic information is posted onto a "notice-board" together with an identifier (usually an E.164 number). This dynamic information can be retrieved through the use of the RETRIEVE CALL INFORMATION SIB (see below).

The CIU SIB consists of two operations: Add Call Information and Remove Call Information. The former posts the call information, while the latter removes it. A time can be specified during which the information is to stay posted after an Add Call Information operation, otherwise, the information stays available until it is removed with the Remove Call Information operation.

This SIB is peculiar in that it is designed to be part (often the only part) of a SIB chain that is only potentially part of a realised service – a service realised only when the appropriate service instance invocation takes place from the Internet.

This SIB, in conjunction with the Retrieve Call Information SIB, can be used for asynchronous communication between independent (not in a parent – child relationship) SIB chains pertaining to the same IIN service instance.

c) Potential service applications

- Parallel Routing
- Fax-on-Demand (with third party-provided special resource)
- Unified Messaging (with third party-provided special resource)
- IVR (with third party-provided special resource)

d) Input

Logical start

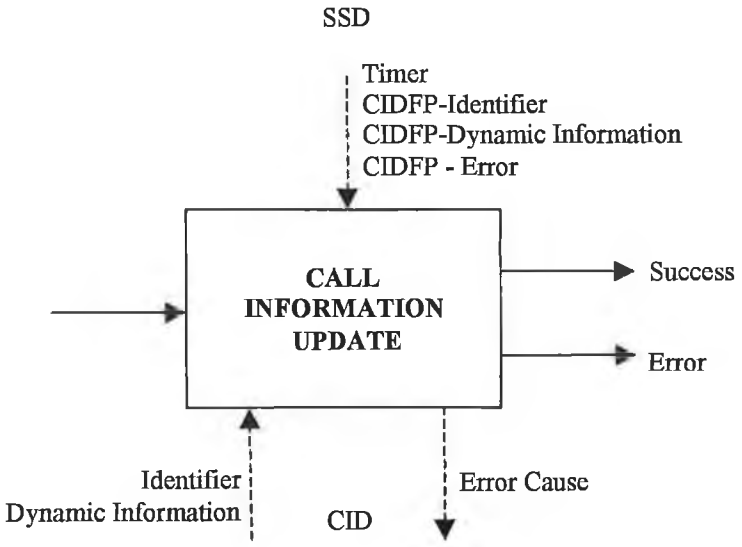
Indicates the logical start of execution for the SIB.

Service Support Data

- Timeout – length of time during which the Call Information should stay available.
- CIDFP-Identifier - Call Instance Data field pointer to the Identifier value.
- CIDFP-Dynamic Information - Call Instance Data field pointer to the Dynamic Information value.
- CIDFP-Error - specifies where in output call instance data the error cause will be written.

Call Instance Data

- Identifier – identifies a resource or entity to which the Dynamic Information pertains.
- Dynamic Information - the current information pertaining to the resource or entity represented by Identifier, which is to be stored for possible use by a service instance.



SSD – Service Support Data
CID – Call Instance Data
CIDFP – Call Instance Data Field Pointer

Figure A.1 – GFP Description of the CALL INFORMATION UPDATE SIB

e) Output

Logical End

- Success
- Error

Call Instance Data

- Error cause – identifies the specific condition that caused an error during the operation of the SIB. The possible errors are:
  - ⇒ invalid identifier value;
  - ⇒ invalid dynamic information value.

f) Graphical representation

The graphical representation of this SIB is given in Figure 4.9.

NON-REAL-TIME VOICE-CHANNEL DATA MANIPULATION (NRT-VCDM)

a) Definition

This SIB does various useful manipulations of non-real-time data carried on the voice channel.

b) Operation

The non-real-time adjective in the name of this SIB describes voice-channel data that, at some stage in the course of a service, has a static form, as opposed to the information stream form used to send the information through the voice channel. The SIB is limited to Internet-storable static forms such as files containing facsimile documents, sound files containing voice-mail messages, HTML documents and emails. The SIB may perform different functions, depending on the function type specified as one of its input parameters. Each function type consists of an Internet communication function and a telephone network communication function. The Internet communication functions are retrieve document and send message. The possible telephone network communication functions are receive fax, send fax, user interaction and receive voice message.

c) Potential service applications

- Group Announcements
- Unified Messaging
- Fax-on-Demand
- Interactive Voice Response
- Request-to-Fax

d) Input

Logical start

Indicates the logical start of execution for the SIB.

Service Support Data

- Function Type – specifies the type of function to be performed using non-real-time voice-channel data. The function type may be send fax (retrieves a specified document from the Internet and sends it to a specified receiving fax number), receive fax (connects to a specified calling number and receives a fax from it, then sends it to a specified receiving address, which could be and email address), subscriber-specific user interaction (retrieves a specified document from the Internet, connects to a specified called or calling number, then using the script laid out in the retrieved document performs user interaction) and receive voice message (performs user interaction and receives a voice message, then converts the message and sends it to a specified receiving address, which could be an email address). It should be noted that the above list is not exhaustive.
- CIDFP-Internet Address – Call Instance Data field pointer to the value of Internet Address.
- CIDFP-Telephone Network Address – Call Instance Data field pointer to the value of Telephone Network Address.
- CIDFP-User Interaction Parameters – Call Instance Data field pointer to the User Interaction Parameters. If the function type does not include IN scripted user interaction, the value of this pointer is ignored.
- CIDFP-Collected Data – Call Instance Data field pointer to the value of Collected Data. If the function type does not include IN scripted user interaction, the value of this pointer is ignored.
- CIDFP-Error – Call Instance Data field pointer to the location for the output CID in the case of an error.

Call Instance Data

- Internet Address – a string, holding some type of address in the Internet, for example a URL or an email address. How the SIB will interpret this address depends on the Function Type value.
- Telephone Network Address – the number of a terminal in the telephone network, normally either a telephone or a fax. The SIB interprets the number depending on the function type.
- User Interaction Parameters – a structure with parameters for the user interaction, if it takes place within the specified function type. The parameters are not listed in

detail since they are very similar to those for the CS1 USER INTERACTION SIB.

e) Output

Logical End

- Success
- Error

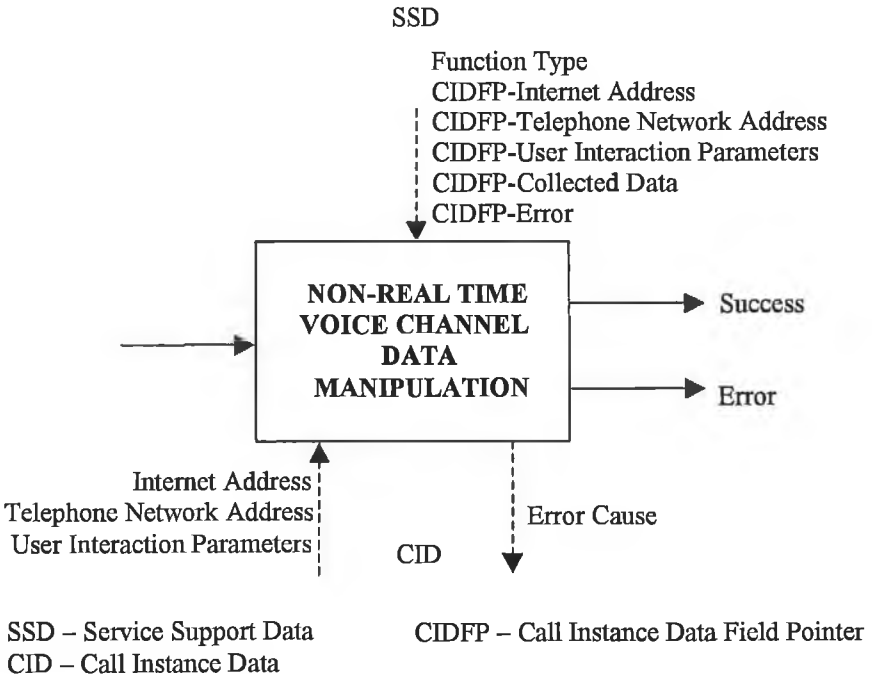


Figure A.2 – GFP Description of the NON-REAL TIME VOICE CHANNEL DATA MANIPULATION SIB

Call Instance Data

- Collected Data – the collected data in the case that user interaction takes place as part of the specified function type.
- Error Cause – identifies the specific condition that caused an error during the operation of the SIB. The possible errors are:
  - ⇒ invalid internet address;
  - ⇒ invalid telephone network address;
  - ⇒ invalid user interaction parameters;
  - ⇒ invalid internet data;
  - ⇒ error in user interaction.

f) Graphical representation

The graphical representation of this SIB can be found in Figure 4.10.

RETREIVE CALL INFORMATION (RCI)

a) Definition

This SIB retrieves dynamic information about some resource, preserved by an instance of the CALL INFORMATION UPDATE SIB.

b) Operation

The dynamic information that this SIB retrieves pertains to some resource in the IN and is not accessible directly from the SIB chain that this SIB is part of. The information is made available by the CALL INFORMATION UPDATE SIB in another SIB chain, which does have access to the information and posts it somewhere whence the RCI SIB can retrieve it. The resources for which the dynamic information is being stored are represented by identifiers, normally E.164 numbers. Such an identifier is used by this SIB to retrieve the dynamic information.

A timeout value can be specified. If this value is present, the SIB will wait for the specified amount of time for the Call Information to be posted by the CIU SIB. If after this time the value is not found, the SIB returns with the Timeout logical end.

Another value that can be specified is the Value to Match, which indicates the value that is expected to be found associated with Identifier. If the identifier is found and the value associated with it differs from Value to Match, the SIB returns with the Failure logical end.

The SIB returns with Failure also in the case that neither Timeout or Value to Match are specified and no Call Information was found, posted with the specified Identifier.

c) Potential service applications

- Parallel Routing
- Fax-on-Demand (with third party-provided special resource)
- Unified Messaging (with third party-provided special resource)
- IVR (with third party-provided special resource)

d) Input

Logical start

Indicates the logical start of execution for the SIB.

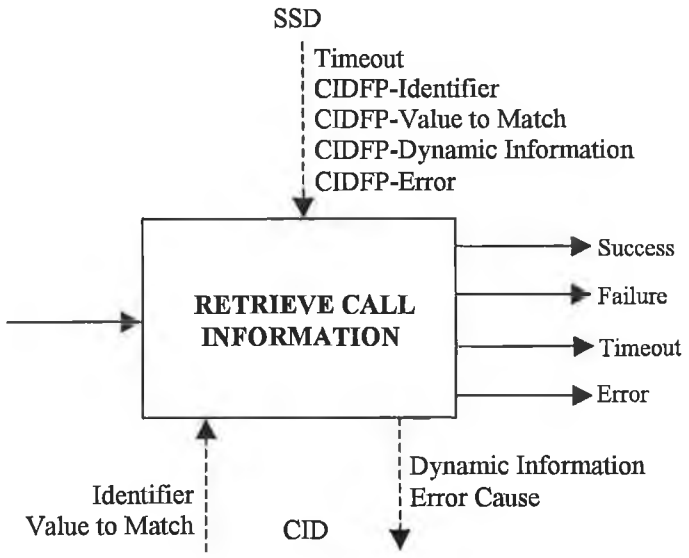
Service Support Data

- Timeout – Specifies the time during which the SIB should keep trying to retrieve information.
- CIDFP-Identifier - Call Instance Data field pointer to the Identifier value.

- CIDFP-Value to Match – Call Instance Data field pointer to the Value to Match value
- CIDFP-Dynamic Information - Call Instance Data field pointer to the location where the output, consisting of the dynamic information, is to be placed.
- CIDFP-Error - specifies where in output call instance data the error cause will be written.

Call Instance Data

- Identifier – identifies a resource or entity for which the dynamic information needs to be retrieved.
- Value to Match – if, present, specifies the value that must be matched as the Call Information. If the Identifier is found but the corresponding value does not match this value, the SIB exits with Failure.



SSD – Service Support Data
CID – Call Instance Data
CIDFP – Call Instance Data Field Pointer

Figure A.3 – GFP Description of the RETRIEVE CALL INFORMATION SIB

e) Output

Logical End

- Success
- Failure
- Timeout
- Error

Call Instance Data

- Dynamic Information - the dynamic information pertaining to the resource or entity represented by Identifier, retrieved by this SIB.
- Error cause – identifies the specific condition that caused an error during the operation of the SIB. The possible errors are:
  ⇒ invalid identifier value.

f) Graphical representation

The graphical representation of this SIB is shown in Figure 4.11.

SEND SERVICE COMPONENT REQUEST (SSCR)

a) Definition

This SIB places a request for an Internet Application Service Component to be executed.

b) Operation

This SIB is used to invoke service components executed outside the basic IN, that is, service components that are not part of the IN service logic. The SIB sends a request to some handling entity, specifying the type of service to be invoked and the necessary parameters.

c) Potential service applications

- Click-to-Dial-Back
- Televoting
- Group Announcements
- Automatic Call Distribution
- Customer Profile Management

d) Input

Logical start

Indicates the logical start of execution for the SIB.

Service Support Data

- Service Component Type - identifies the type of service component for which a request for invocation is being issued by the SIB.
- CIDFP-Address – Call Instance Data field pointer to the Address data field.
- CIDFP-Service Component Parameters - Call Instance Data field pointer to the Service Component Parameter Structure.

- CIDFP-Error - specifies where in output call instance data the error cause will be written.

Call Instance Data

- Address – the Internet Address of the receiver of the Service Component Request. The address contains the IP address and, optionally, the port number.
- Service Component Parameters - an entity containing the parameters that need to be specified for the service component instance by the SIB.

e) Output

Logical End

- Success
- Error

Call Instance Data

- Error cause – identifies the specific condition that caused an error during the operation of the SIB. The possible errors are:
  - ⇒ invalid service component type;
  - ⇒ invalid service component parameters.

f) Graphical representation

The graphical representation of this SIB can be found in Figure 4.12.

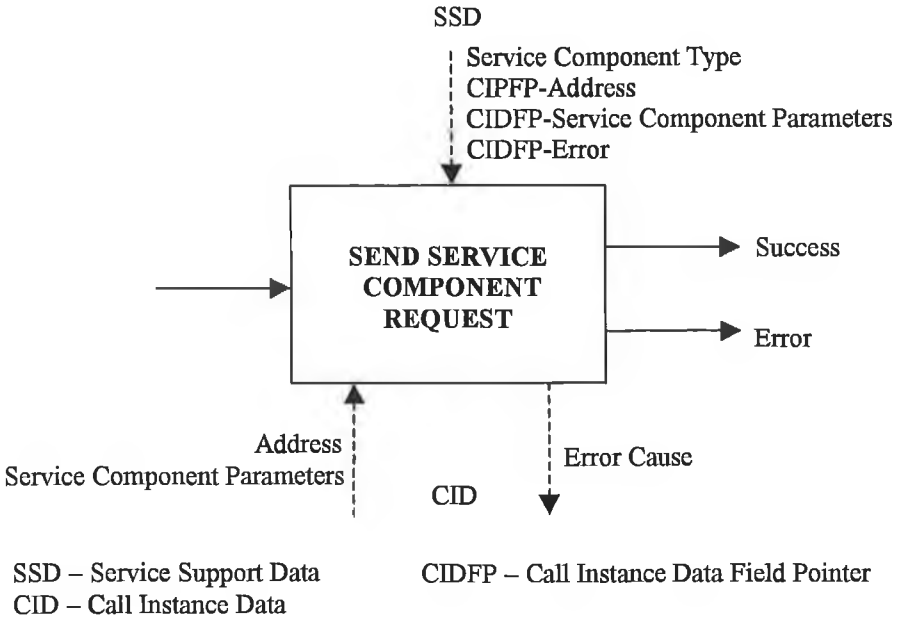


Figure A.4 – GFP Description of the SEND SERVICE COMPONENT REQUEST SIB

TIMER

a) Definition

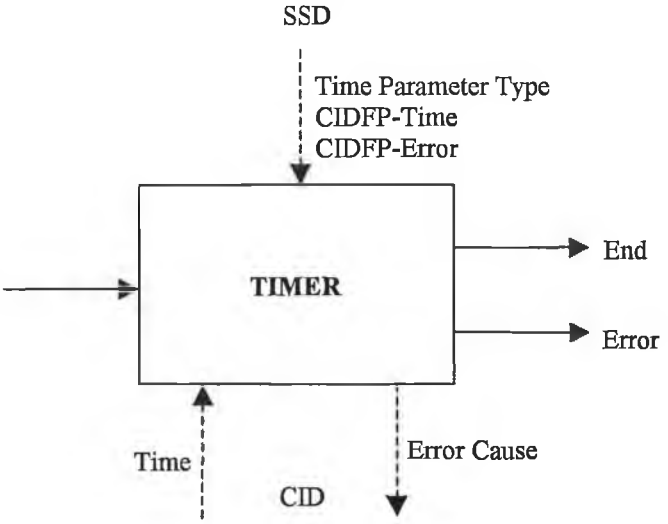
Delays the execution of the SIB chain by a specified time.

b) Operation

This SIB suspends the operation of the SIB chain for a certain length of time. However, this SIB is not woken up by an outside stimulus but by an internal timer. At its invocation the SIB is given either a length of time for which it is to suspend the processing or a moment in the future when it is to end the suspension. Once it wakes up, the SIB ends its operation, signalling implicitly that it is time for the next SIB in the chain to be invoked.

c) Potential service applications

- Click-to-Dial-Back
- Alarm Call



SSD – Service Support Data
CID – Call Instance Data
CIDFP – Call Instance Data Field Pointer

Figure A.5 – GFP Description of the TIMER SIB

d) Input

Logical start

Indicates the logical start of execution for the SIB.

Service Support Data

- Time parameter type - specifies whether the time parameter is absolute i.e. a moment in the future, or relative i.e. a length of time to wait.
- CIDFP- Time - Call Instance Data field pointer to the Time value.

- CIDFP-Error - specifies where in output call instance data the error cause will be written.

Call Instance Data

- Time – time value to be used by the timer, can be either absolute or relative.

e) Output

Logical End

- End
- Error

Call Instance Data

- Error cause – identifies the specific condition that caused an error during the operation of the SIB. The possible errors are:
  - ⇒ invalid time value.

f) Graphical representation

The graphical representation for this SIB is in Figure 4.13.

Appendix B

Stage 2 Description of the IICSI1 SIBs

In following the convention for numbering Functional Entity Actions (FEA) with XYYZ, where X is a digit representing a Functional Entity, YY a number representing a SIB and Z a digit representing the FEA within the SIB, the new entities have been assigned as follows:

- C (instead of a digit) for the ISCGF
- R (instead of a digit) for the ISRGF
- i1, i2, i3 etc. (instead of two digit numbers) for the FEA numbers

CALL INFORMATION UPDATE

a) Description

This SIB updates call information needed by some IIIN service. The SIB is always part of an SLP which was invoked as a result of an initiating Information Flow (IF) from the SSF, such as the Initial DP IF or a DP-specific IF sent at a Trigger Detection Point (TDP) in the BCSM. The information kept in this way will be used by an SLP belonging to the same service but invoked from the Internet-SCF Gateway. The SIB is performed entirely in the SCF.

b) Information Flows

No IFs are required for this SIB.

c) SDLs

The SDL description of the functionality of this SIB is in Figure 4.15.

d) Functional Entity Actions

- Perform call information update - creates, updates or destroys call information for a particular call. Performed by the SCF (9i11).

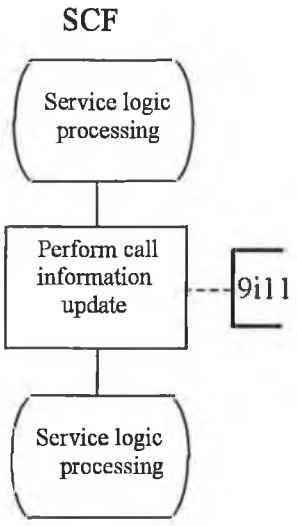


Figure B.1 – SDL Representation of CALL INFORMATION UPDATE SIB Functionality

NON-REAL-TIME VOICE-CHANNEL DATA MANIPULATION

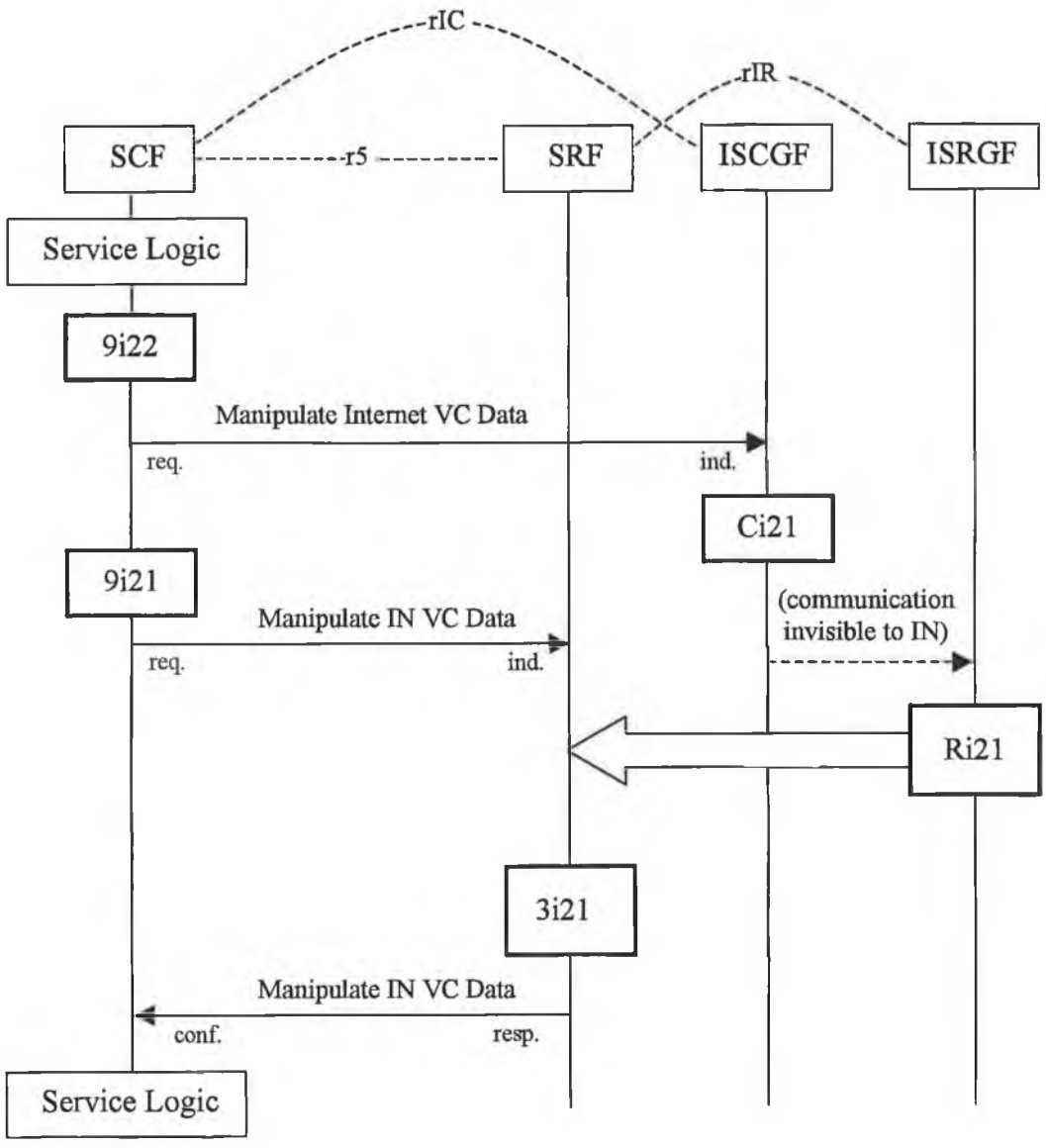
a) Description

This SIB performs the conversion and transport between the Internet and the telephone network of voice-channel data, in either direction. It may retrieve voice-channel data from the Internet, in an Internet-native format, convert it to a telephone-network-native format, and send it to a specified address in the telephone network. Or it may receive voice-channel data from the telephone network, in a telephone-network-native format, convert it to a Internet-native format, and send it to a specified address in the Internet. The SIB makes use of a Special Resource Function (SRF) in the IN and the Internet-SCF Gateway Function (ISCGF). First a request is sent from the SCF to the SSF to connect a call party to the SRF, then a request is sent from the SCF to the SRF to perform the operation. The operation includes a request from the SRF to the Internet-SRF Gateway for Internet communication.

b) Information Flows

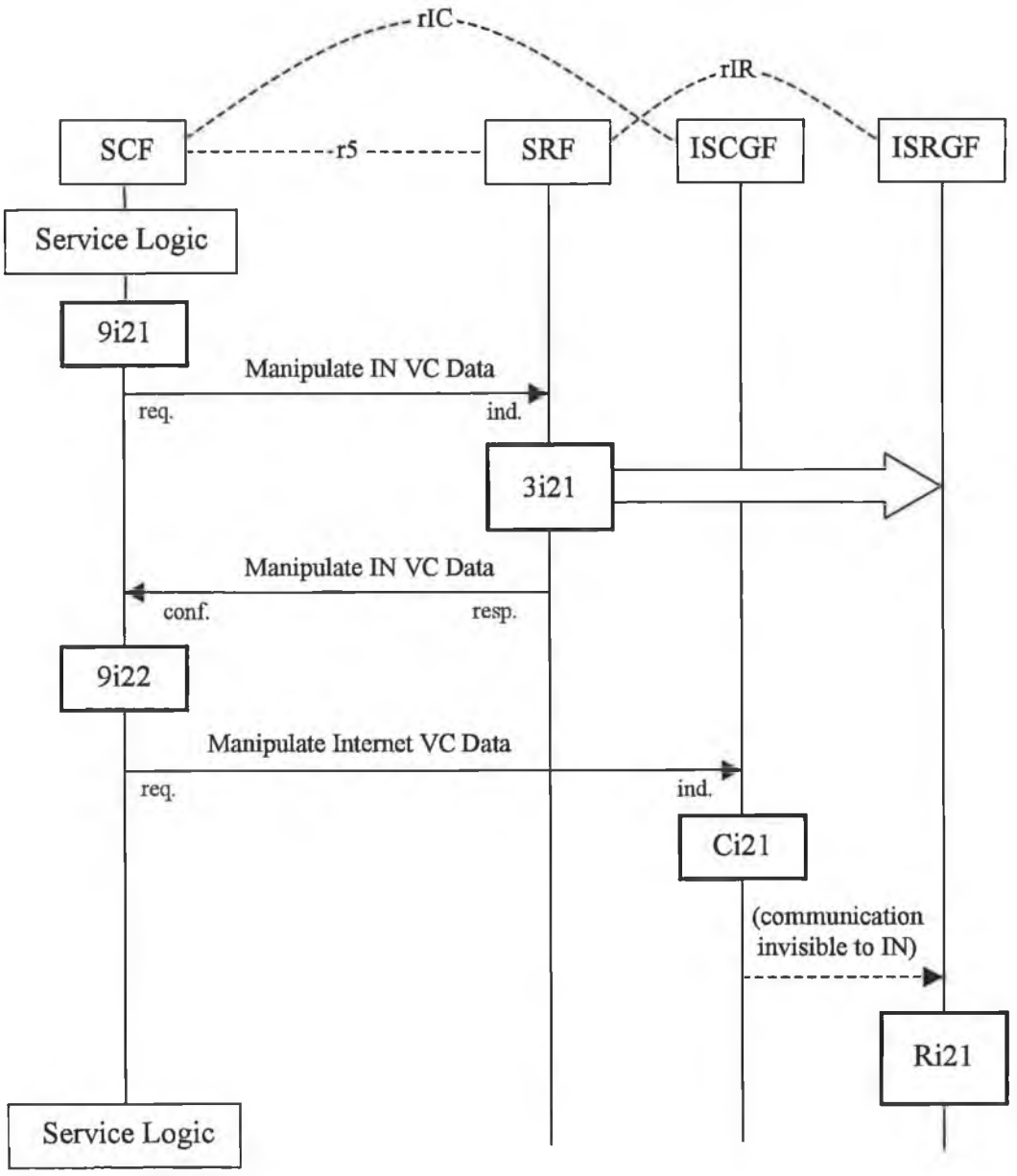
Diagrams

Figure 4.16 shows a Message Sequence Charts (MSCs) of the IFs and FEAs pertaining to this SIB, for the case where the flow of voice-channel data is in the direction from the Internet to the IN and vice versa. The MSCs show the case with no errors occurring during operation.



SCF – Service Control Function
SRF – Special Resource Function
ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
VC Data – Voice-Channel Data

Figure B.2 a) – Information Flow Diagram for the NON-REAL TIME VOICE-CHANNEL DATA MANIPULATION SIB
(in the case of data flow from the Internet to the IN)



SCF – Service Control Function
SRF – Special Resource Function
ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
VC Data – Voice-Channel Data

Figure B.2 b) – Information Flow Diagram for the NON-REAL TIME VOICE-CHANNEL DATA MANIPULATION SIB
(in the case of data flow from the IN to the Internet)

Definition of IFs

1) *Manipulate IN Voice-Channel Data* request (indication)

IEs:

- CallPartyAddress (mandatory)
- OperationType (optional)
- UIParameters (optional)

2) *Manipulate Internet Voice-Channel Data* request (indication)

IEs:

- OperationType (mandatory)
- InternetAddress (mandatory)

3) *Voice-Channel Data* data

IEs:

- Data

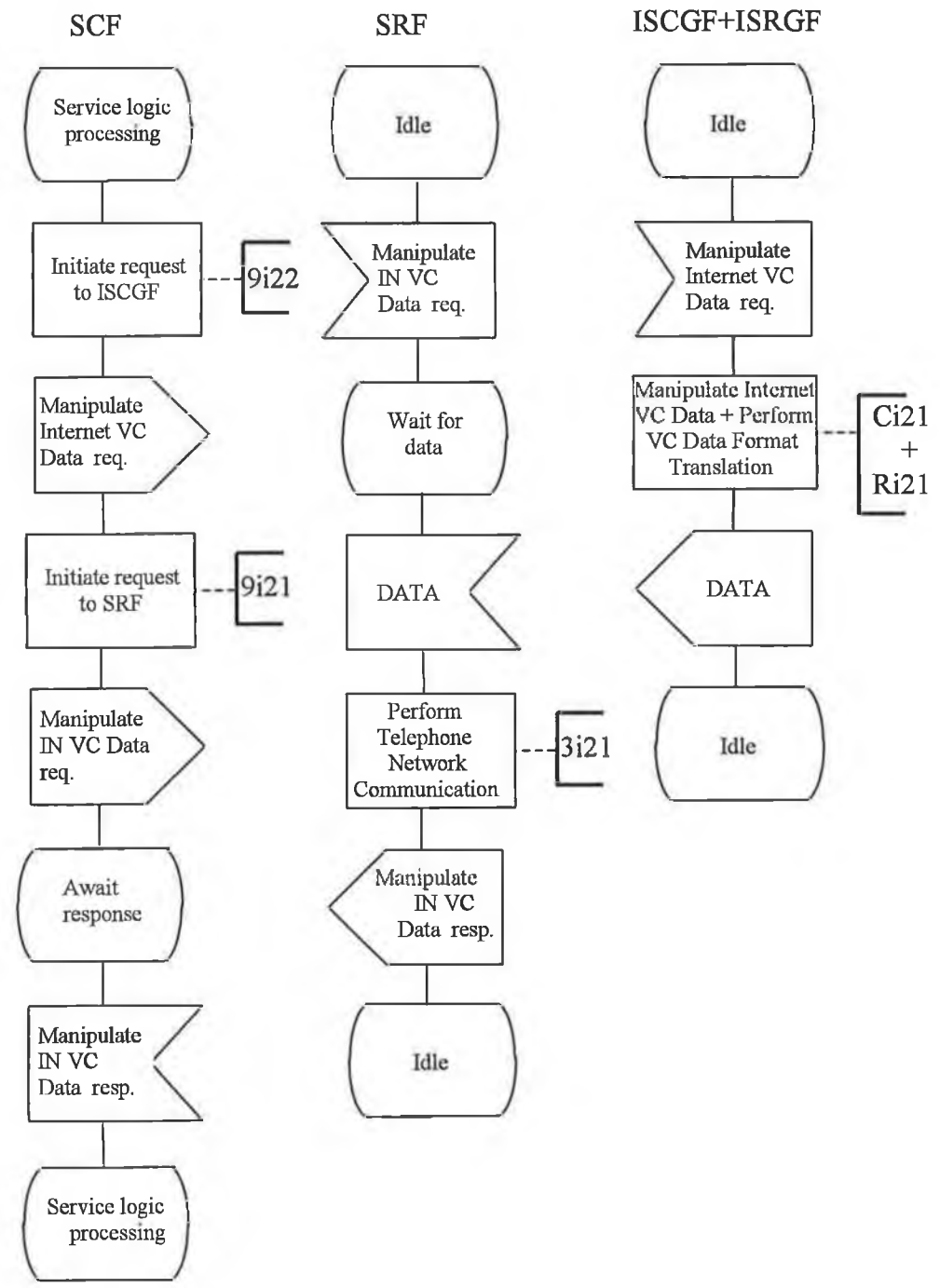
c) SDL

The SDL diagrams representing the distributed functionality of this SIB can be found in Figure 4.17.

d) Functional Entity Actions

- Initiate Request to SRF - prepares and sends the Manipulate IN Voice-Channel Data IF. Performed by the SCF (9i21).
- Initiate Request to ISCGF – prepares and sends the Manipulate Internet Voice-Channel Data IF. Performed by the SCF (9i22).
- Perform Telephone Network Communication - performs user interaction including the receipt or sending of a voice message, sends a fax or receives a fax. This FEA also includes the receiving or sending of data to the ISRGF for conversion, either simultaneously with the communication or just before/after the communication. Performed by the SRF (3i21).
- Manipulate Internet Voice-Channel Data – sends a service request related to the ISRGF to the Internet. Performed by the ISCGF (Ci21).
- Perform Voice-Channel Data Format Translation – translates the voice-channel data from an IN-native format to an Internet-native format. Performed by the ISRGF (Ri21).

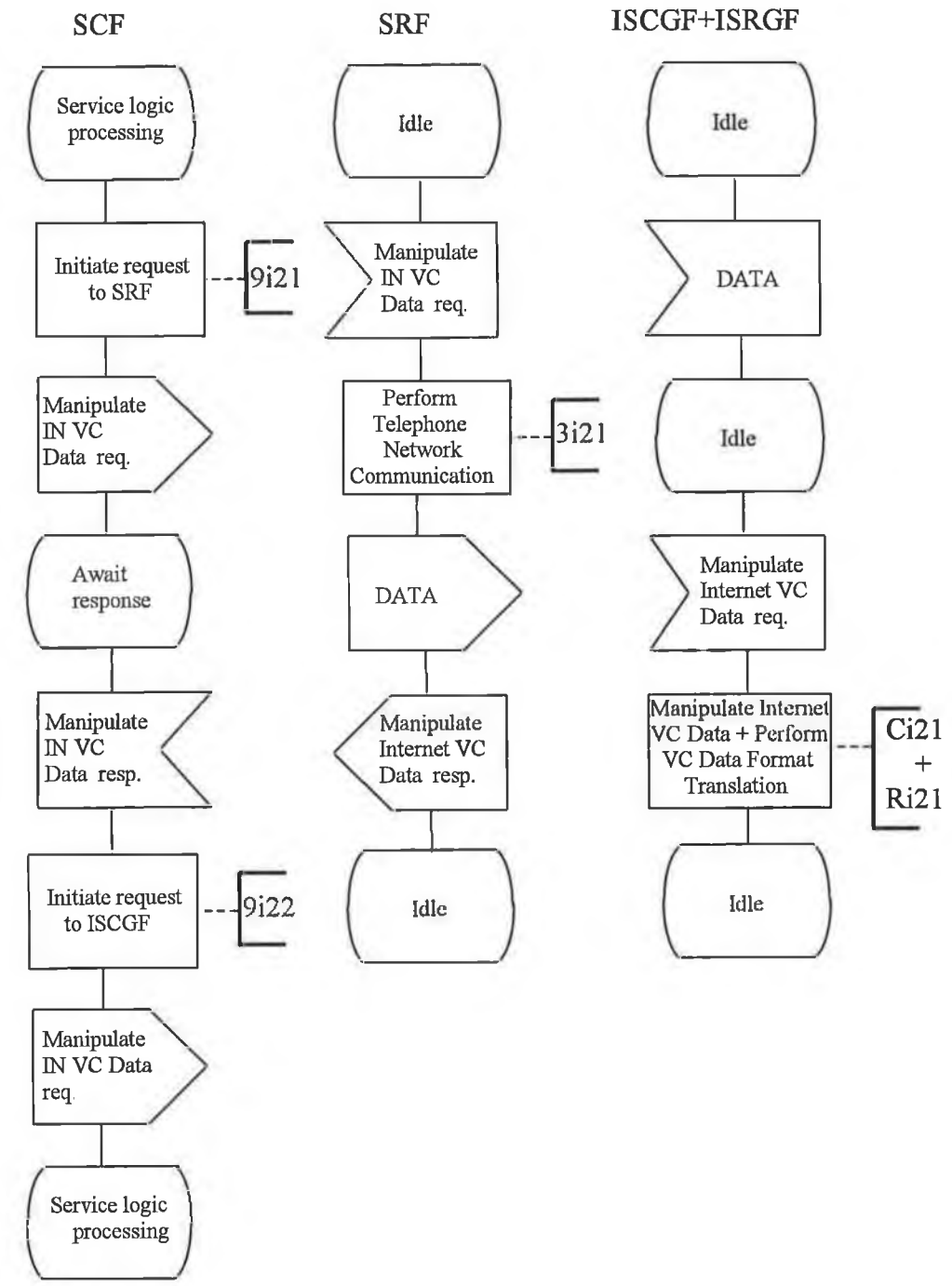
Note1: Depending on the operation type, the FEAs are performed in the SRF and the Internet-SRF Gateway in different orders.



SCF – Service Control Function
SRF – Special Resource Function

ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
VC Data – Voice-Channel Data

Figure B.3 a) – SDL Representation of NON-REAL-TIME VOICE CHANNEL DATA MANIPULATION SIB Functionality
(in the case of data flow from the Internet to the IN)



SCF – Service Control Function
SRF – Special Resource Function

ISCGF – Internet-SCF Gateway Function
ISRGF – Internet-SRF Gateway Function
VC Data – Voice-Channel Data

Figure B.3 b) – SDL Representation of NON-REAL-TIME VOICE CHANNEL DATA MANIPULATION SIB Functionality
(in the case of data flow from the IN to the Internet)

RETREIVE CALL INFORMATION

a) Description

This SIB retrieves call information stored by the UPDATE CALL INFORMATION SIB. It is used in SLPs invoked from the Internet-SCF Gateway.

b) Information Flows

No IFs are required for this SIB.

c) SDLs

The SDL diagram describing the functionality of this SIB is in Figure 4.18

d) Functional Entity Actions

- Perform call information retrieval - retrieves some previously stored call information. Performed by the SCF (9i31).

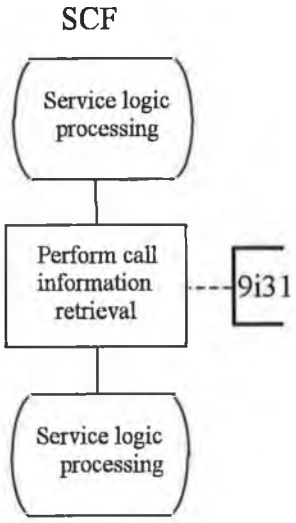


Figure B.4 – SDL Representation of RETRIEVE CALL INFORMATION SIB Functionality

SEND SERVICE COMPONENT REQUEST

a) Description

This SIB sends a request for the execution for an Internet Application Service Component (IASC) and the IASC is executed by Internet resources. This SIB is used in SLPs invoked from the SSF.

b) Information Flows

Diagrams

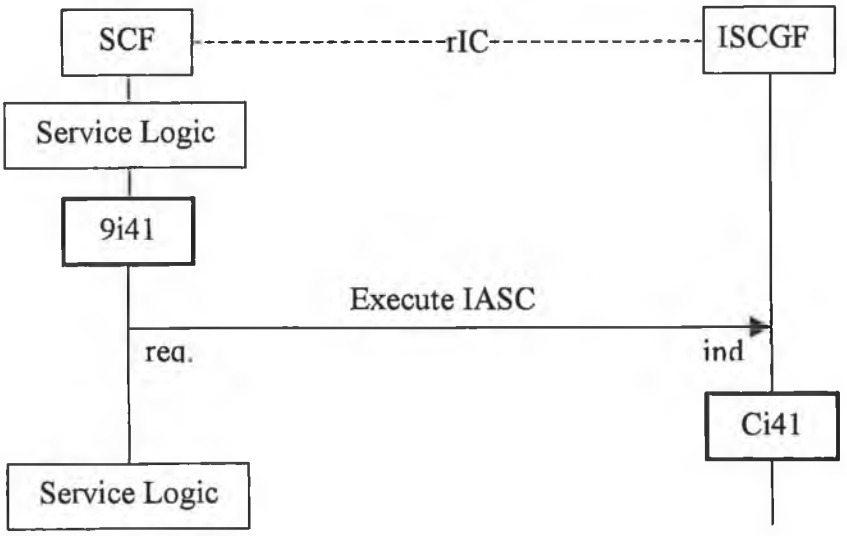
Figure 4.19 shows a Message Sequence Chart of the IFs and FEAs pertaining to this SIB, for the case of normal operation, without errors.

Definition of IFs

1) Execute Internet Application Service Component request (indication)

IFEs:

- ServiceComponentType (mandatory)
- ServiceComponentParameters (optional)



SCF – Service Control Function
ISCGF – Internet-SCF Gateway Function
IASC – Internet Application Service Component

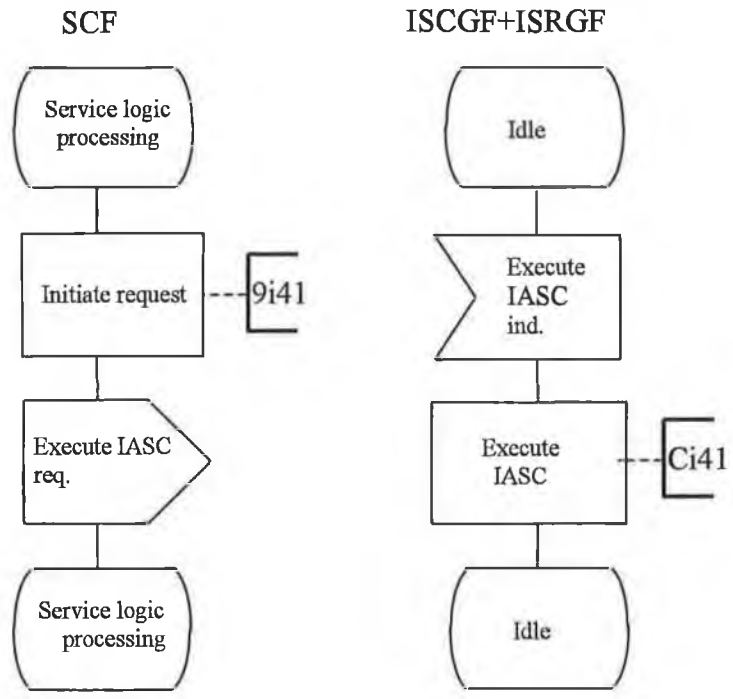
Figure B.5 – Information Flow Diagram for the SEND SERVICE COMPONENT REQUEST SIB

c) SDLs

The SDL description of this SIB can be found in Figure 4.20.

d) Functional Entity Actions

- Initiate Request - prepares and sends the Execute Internet Application Service Component IF. Performed by the SCF (9i41).
- Execute IASC - executes the requested IASC. Performed by the Internet-SCF Gateway (Ci41).



SCF – Service Control Function
ISCGF – Internet-SCF Gateway Function
IASC – Internet Application Service Component

Figure B.6 – SDL Representation of SEND SERVICE COMPONENT REQUEST SIB Functionality

TIMER

a) Description

This SIB sets a timer to a certain value and once the timer expires, ends its own operation. This SIB can be used both in SLPs invoked by the SSF and SLPs invoked by the Internet-SCF Gateway.

b) Information Flows

No IFs are required for this SIB.

c) SDLs

The SDL diagram of this SIB's functionality is in Figure 4.21.

d) Functional Entity Actions

- Set timer and wait for expiration - sets a timer and returns as soon as that timer expires. Performed by the SCF (9i51).

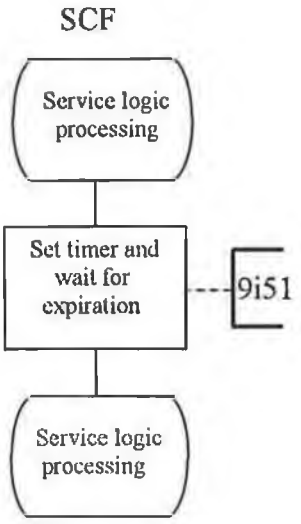


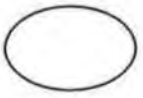
Figure B.7 – SDL Representation of TIMER SIB Functionality

Appendix C

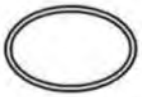
SIB and IASC Chains Representing IICSI1 IIIN Services

This Appendix contains figures that describe how the services provided by the Internet Integrated Intelligent Network (IIIN) with capabilities of the Internet-Integration Capability Set Increment 1 (IICSI1) are composed of Service-Independent Building Blocks (SIBs), Internet Application Service Components (IASCs) and, in some cases, Service-Dependent Building Blocks (SDBs).

The following graphical symbols are used:



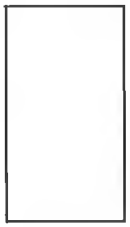
SIB



SDB

Call Initiation (not a SIB but is an IN capability)

IASC



Basic Call Process, with a description of user at side at which the service logic initiating Detection Point is

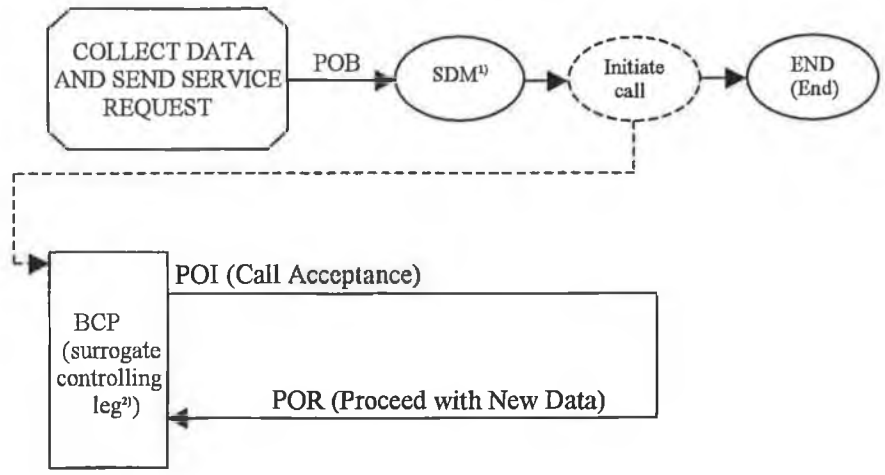
Service logic flow and inter-SIB communication, depending on context

BCP or SIB-chain creation

The following acronyms are used:

BCP	Basic Call Process
POI	Point of Initiation
POR	Point of Return
UI	User Interaction
SDM	Service Data Management
MH	Message Handler
POB	Point of Beginning
POE	Point of Extension
SSCR	Send Service Component Request
CIU	Call Information Update
RCI	Retrieve Call Information
NRT-VCDM	Non-Real-Time Voice Channel Data Manipulation
ACD	Automatic Call Distribution

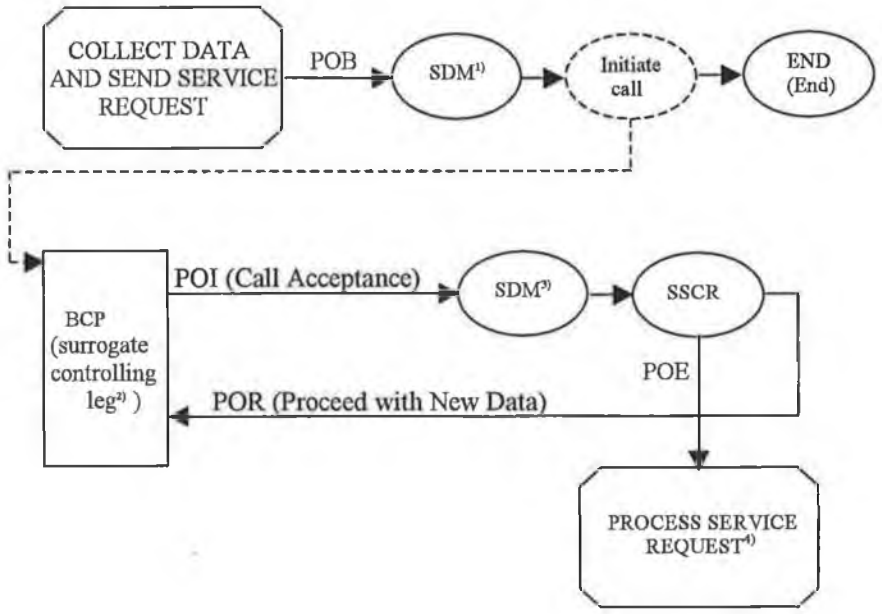
Apart from generally demonstrating the IICSI1 Service functionality, the figures show, for the services that have been implemented as part of the IIIN prototype, how implemented SIBs and IASCs have been chained together to implement the services.



NOTES: 1) Resolves the number that is to be called, based on the IP address of the requestor; this action simultaneously serves as a check as to whether the requesting user is a registered IIIN user

2) As defined in [Q1224]. The flow of events in this BCP corresponds to the following sequence of Call View States (these are described in [Q1224]): *Null -> 1-Party Setup -> Stable 1-Party -> Forward -> Transfer*

Figure C.1 – SIB & IASC Chains for the Click-to-Dial Service



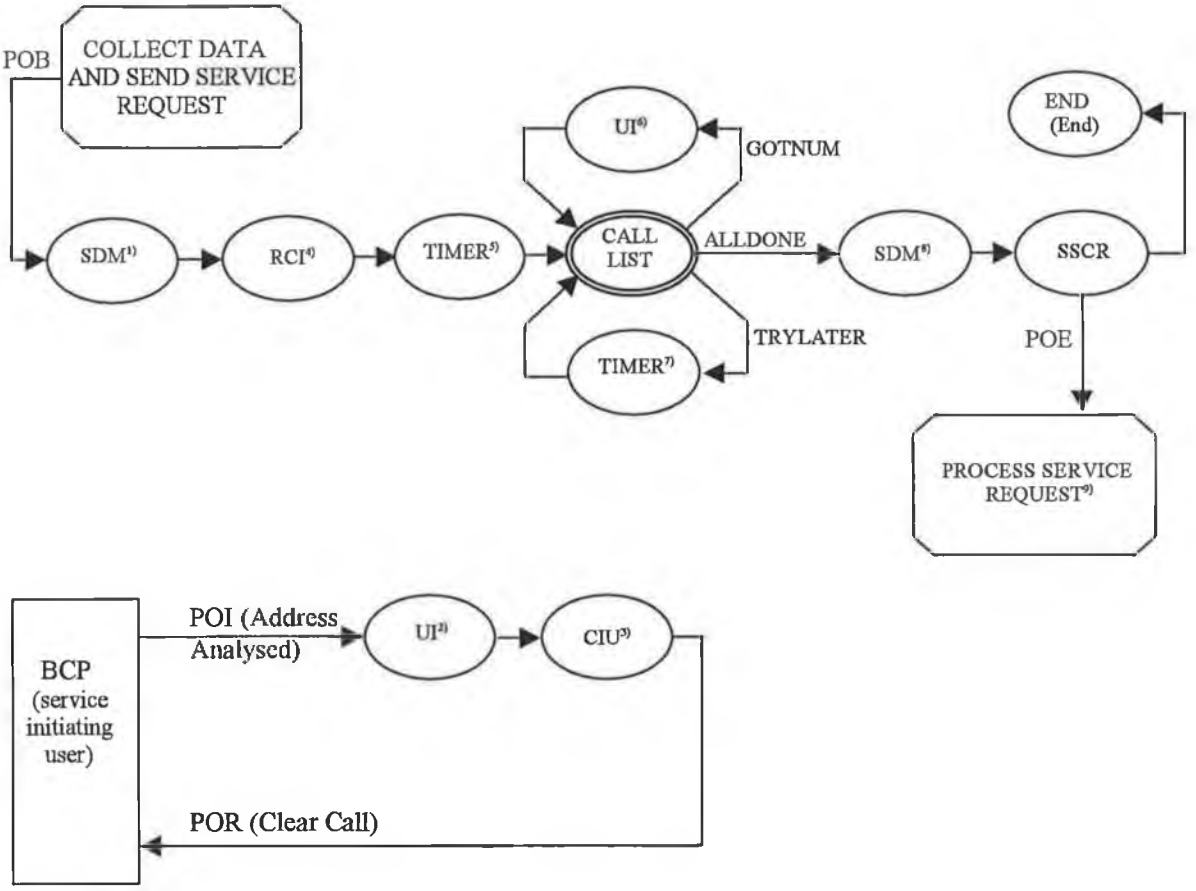
NOTES: 1) Resolves the number that is to be called, based on the IP address of the requestor; this action simultaneously serves as a check as to whether the requesting user is a registered IIIN user

2) As defined in [Q1224]. The flow of events in this BCP corresponds to the following sequence of Call View States (these are described in [Q1224]): *Null -> 1-Party Setup -> Stable 1-Party -> Forward -> Transfer*

3) Finds the IP address of the agent

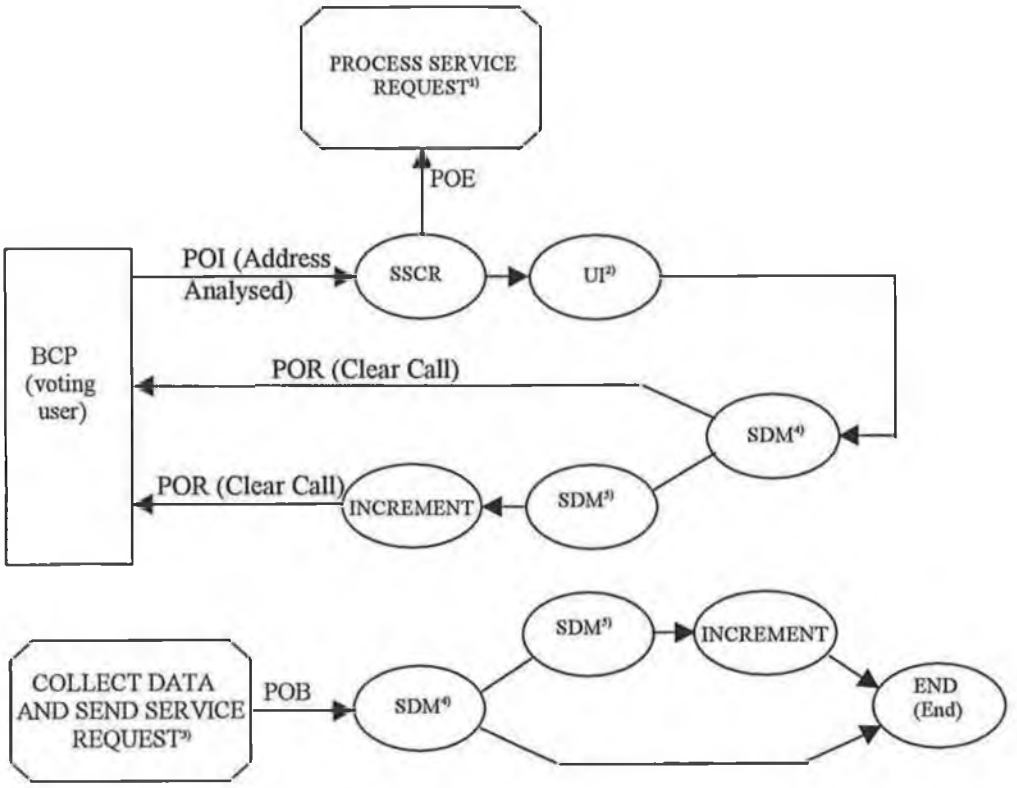
4) Pop-up with service-initiating user's data on agent's screen

Figure C.2 – SIB & IASC Chains for the Click-to-Dial-Back Service



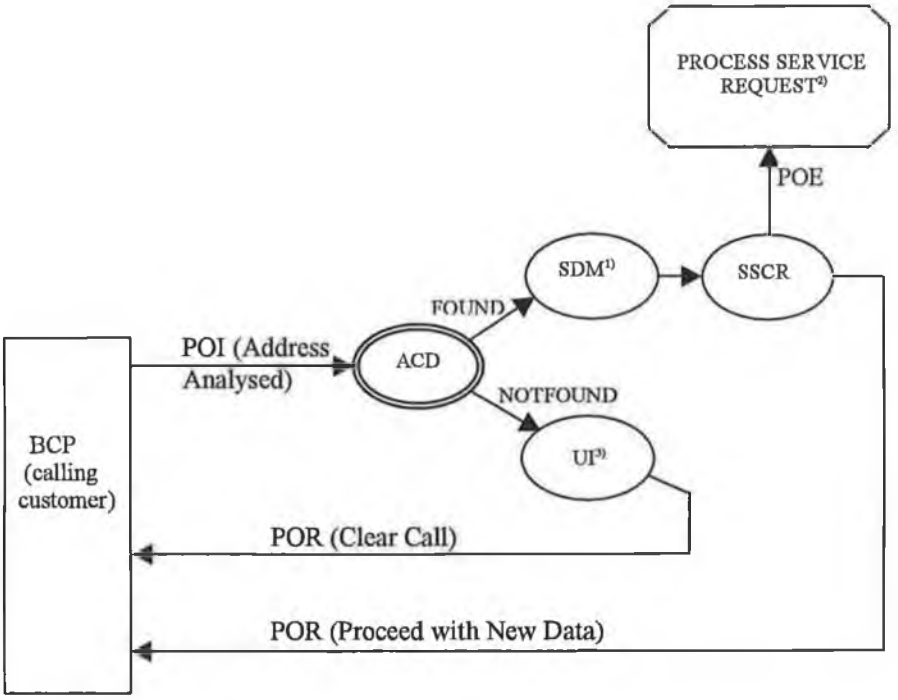
NOTES: 1) Resolves the number that is to be called, based on the IP address of the requestor; this action simultaneously serves as a check as to whether the requesting user is a registered IIIN user
2) Records the announcement by the service-initiating user
3) Posts a reference to the announcement so that the RCI SIB can retrieve it
4) Retrieves the reference to the announcement and stored by the CIU SIB in the chain started from the BCP
5) Suspends the SIB chain until the time to make the announcement comes
6) Plays announcement from service-initiating user to a number on the list
7) Suspends the SIB chain until the time when the announcing should be retried to the numbers for which it has not been successful the previous time round
8) Gets the e-mail address of the initiating user
9) Sends e-mail with the service results to the service-initiating user

Figure C.3 – SIB & IASC Chains for the Group Announcements Service



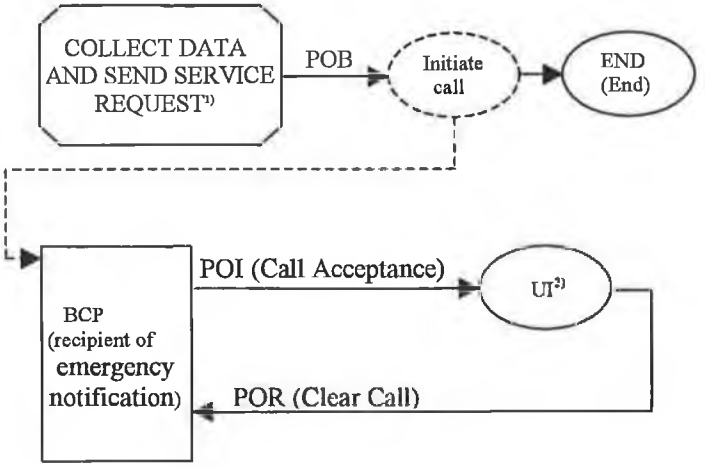
NOTES: 1) Pops-up tele-voting information on the voting user's screen
2) Collects the vote of the voting user by digit detection
3) The collection of a vote from the Internet (most likely from the Web page popped-up in 1)
4) Looks for the voting user's number on the list of numbers that already voted
5) Adds the voting user's number to the list of numbers that already voted

Figure C.4 – SIB & IASC Chains for the Tele-Voting Service



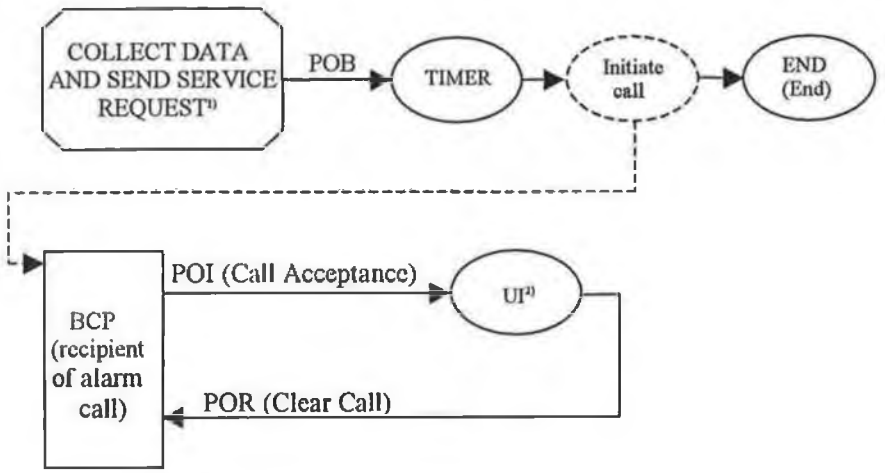
NOTES: 1) Finds data related to the calling customer
2) Pops-up the data related to the calling customer on the agent's screen
3) Plays an announcement to the calling customer to say that there no agent is available to take the call

Figure C.5 – SIB & IASC Chain for the Automatic Call Distribution Service



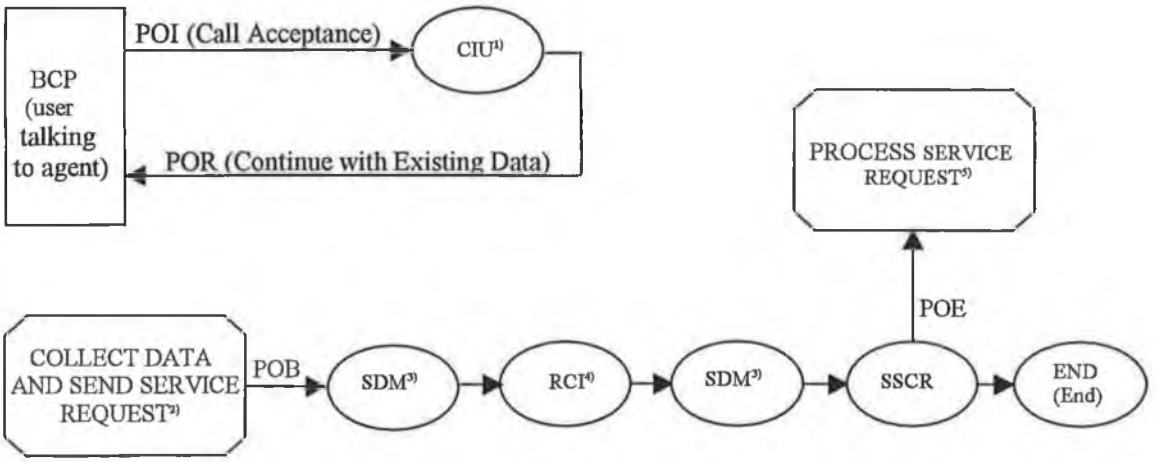
NOTES: 1) A sensor connected to the Internet has detected an emergency condition
2) Announcement informing the recipient of the emergency condition

Figure C.6 – SIB & IASC Chains for the Emergency Notification Service



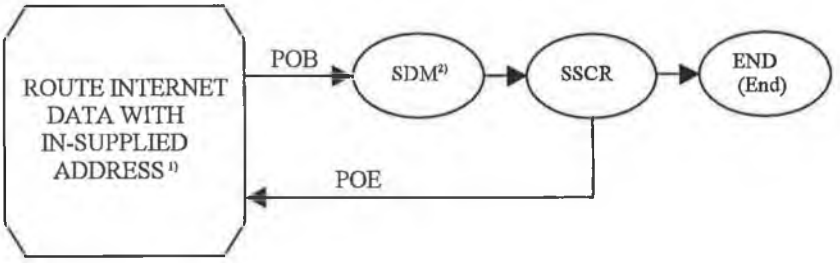
NOTES: 1) The time when the alarm call is to occur is collected from the user
2) Alarm call announcement is played to the recipient

Figure C.7 – SIB & IASC Chains for the Alarm Call Service



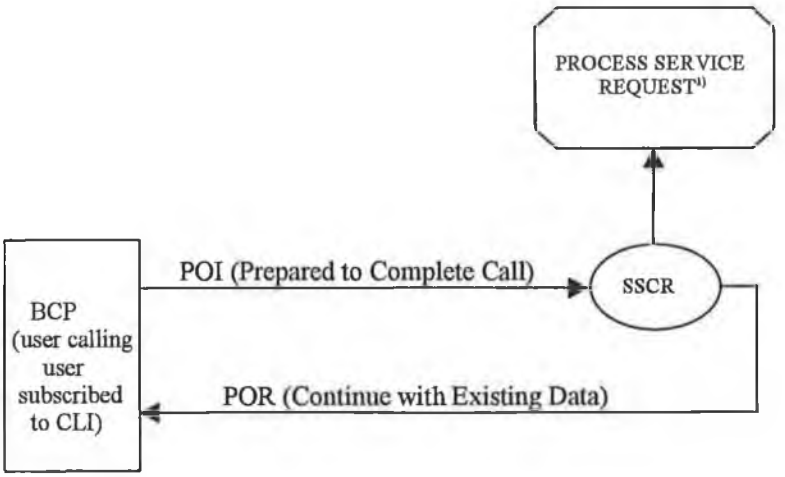
NOTES: 1) The telephone number of the user talking to the agent is noted
2) The agent chooses the data to be sent from a Web page and submits a request for the data to be sent
3) Resolves the number that is to be called, based on the IP address of the requestor; this action simultaneously serves as a check as to whether the requesting user is a registered IIIN user
4) The telephone number of the user talking to the agent is retrieved
5) The IP address of the user talking to the agent is acquired, based on the telephone number retrieved in y the RCI SIB in 4)
6) The data sent by the agent is popped-up on the screen of the user talking to the agent

Figure C.8 – SIB & IASC Chains for the Parallel Routing Service



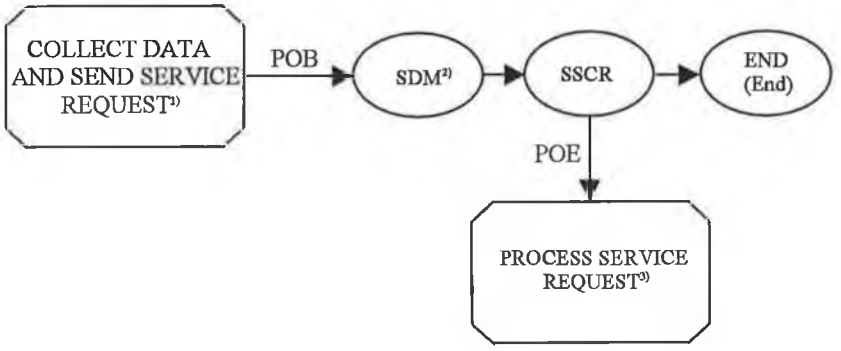
NOTES: 1) An e-mail is sent with a telephone number specified instead of the e-mail address. This IASC sends a request for service to the IN, which results in a return request from the IN, containing the needed e-mail address
2) The e-mail address is retrieved, based on the telephone number

Figure C.9 – SIB & IASC Chains for the Email-by-Number Service



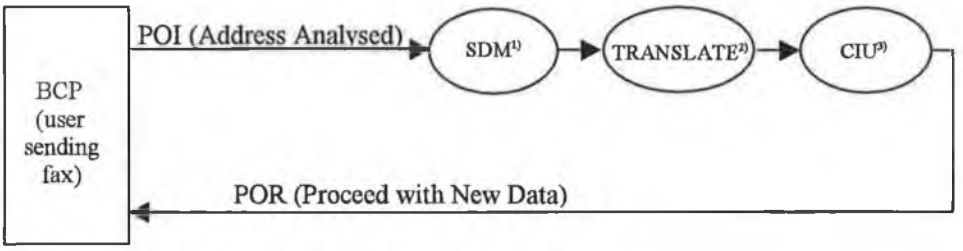
NOTES: 1) Pops-up the number of the calling user on the screen of the user to whom the call is about to be completed (and who must be subscribed to the CLI service)

Figure C.10 – SIB & IASC Chain for the Calling Line Identification Service

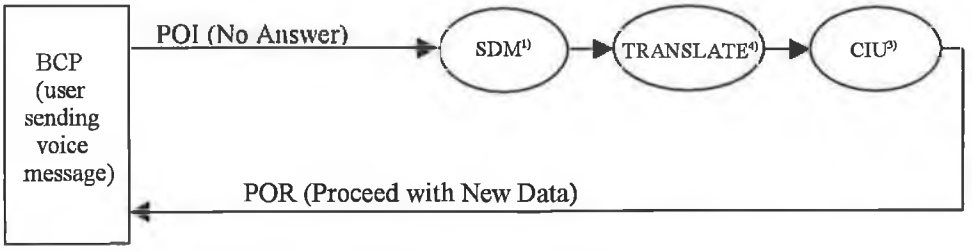


NOTES: 1) The user enters data for the CPM service feature
2) The data pertaining to a service, arrived in the POB, is added to the database entry for the requesting user
3) An acknowledgement of successful completion is popped-up on the user's screen

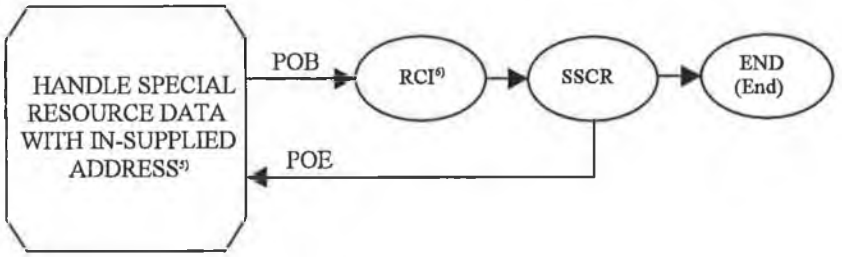
Figure C.11 – SIB & IASC Chain for the Customer Profile Management Service Feature



a) IN part of the service when receiving a fax



b) IN part of the service when receiving a voice message



c) Internet part of the service for both cases

NOTES: 1) Using the dialled number (which is a "virtual" fax number in the case of receiving fax and the user's telephone number in the case of receiving a voice message), finds the destination e-mail address

2) Translates the dialled number into a physical address of a fax-receiving third party Special Resource Device in the telephone network

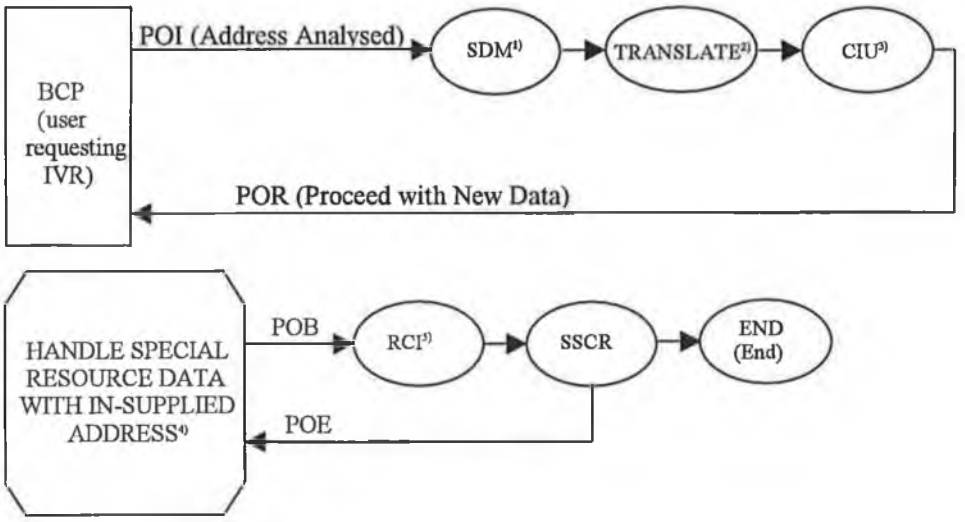
3) Stores the e-mail address as dynamic call information for the physical address

4) Translates the dialled telephone number to the number of a voice-message-receiving third party Special Resource Device in the telephone network

5) Activity of the third party Special Resource Controller at the time of receiving data. Sends a request for IN service component, then waits for a request in the other direction, which contains the destination address. This activity should take place when the 3pSRC has been connected to the user sending the fax or voice message but has not yet answered the call. It should answer the call only once it has the address – in that way an address is matched with a call with complete certainty.
Once the fax is received or message recorded and the format conversion is finished, the data is sent to the e-mail address.

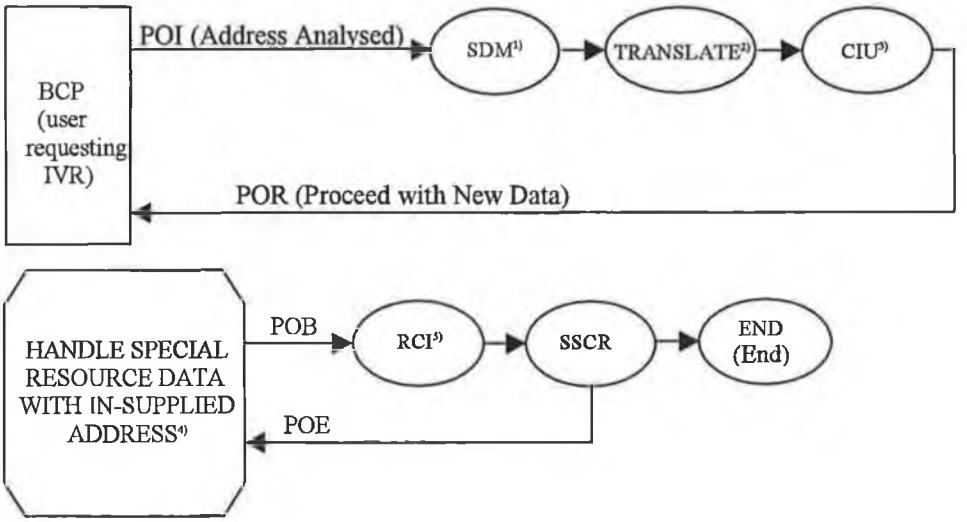
6) Retrieves the destination address (earlier stored by the CIU SIB)

Figure C.12 – SIB & IASC Chains for the Unified Messaging Service When Provided by a Third Party Special Resource

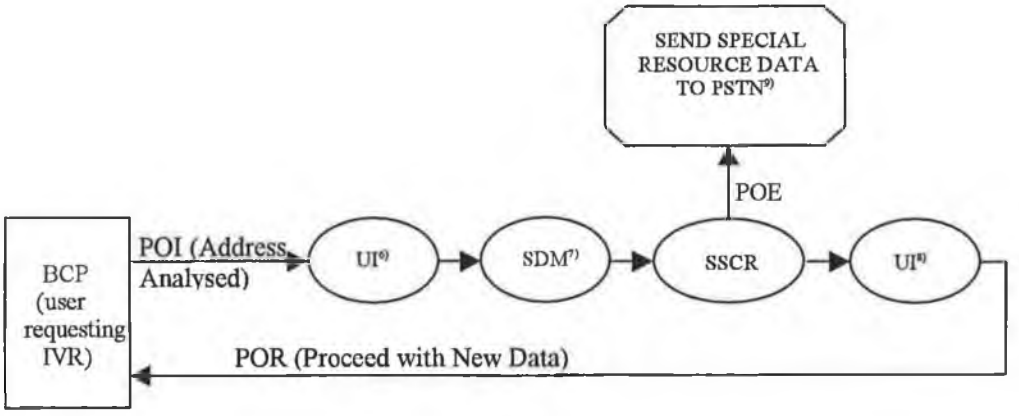


NOTES: 1) Using the dialled number, finds the URL containing the IVR script
2) Translates the dialled number into a physical address in the telephone network
3) Stores the URL as dynamic call information for the physical address
4) Activity of the third party Special Resource Controller at the time of executing the IVR script. First a request for an IN service component is sent. Then the 3pSRC waits for a request in the other direction, which contains the URL of the IVR script to be executed. This activity should take place when the 3pSRC has been connected to the user requesting IVR, but has not yet answered the call. It should answer the call only once it has the URL – in that way a URL is matched with a call with complete certainty.
5) Retrieves URL of the IVR script to be executed, stored earlier in the CIU SIB

Figure C.13 – SIB & IASC Chains for the Interactive Voice Response (IVR) Service When Provided by a Third Party Special Resource



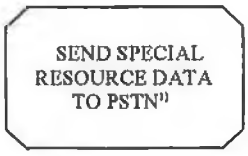
a) Case where the document to be faxed is chosen through IVR



b) Case where the document to be faxed is chosen through IN User Interaction

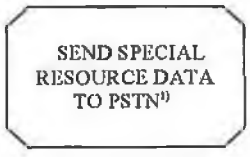
NOTES: 1) Using the dialled number, finds the URL containing the IVR script
2) Translates the dialled number into a physical address in the telephone network
3) Stores the URL as dynamic call information for the physical address
4) Activity of the third party Special Resource Controller at the time of executing the IVR script to determine the identity of the document to be faxed. First a request for an IN service component is sent. Then the 3pSRC waits for a request in the other direction, which contains the URL of the IVR script to be executed. This activity should take place when the 3pSRC has been connected to the user requesting IVR, but has not yet answered the call. It should answer the call only once it has the URL – in that way a URL is matched with a call with complete certainty. The result of the IVR interaction should indicate the document to be sent. The 3pSRC proceeds to fetch and send the document to an address in the telephone network.
5) Retrieves URL of the IVR script to be executed, stored earlier in the CIU SIB
6) Through digit detection reads the code of document to be faxed and destination fax number
7) Determines the URL of the document to be faxed from the document code
8) Announcement to user after successfully completed service
9) Fetches the document from the URL and faxes the document through the 3pSRD

Figure C.14 – SIB & IASC Chains for the Fax-on-Demand Service When Provided by a Third Party Special Resource



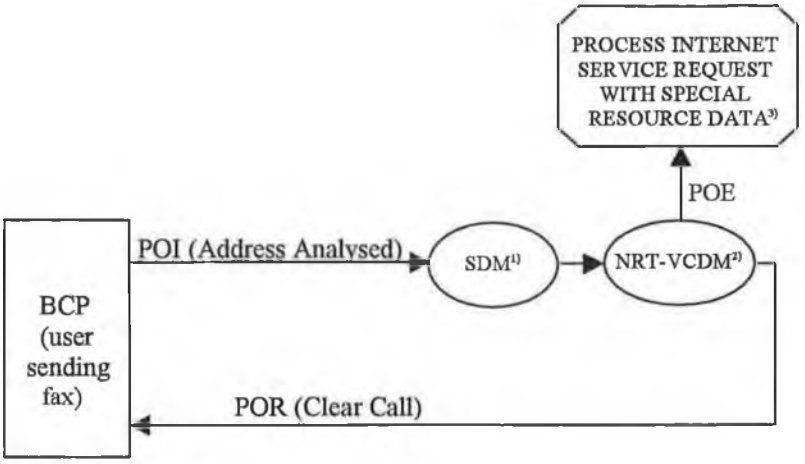
NOTES: 1) Collects information about the URL containing the document to be faxed and the destination fax number. Then sends a request to the 3pSRC to fax the document at that URL to the destination fax number.

Figure C.15 – SIB & IASC Chains for the Request-to-Fax Service When Provided by a Third Party Special Resource

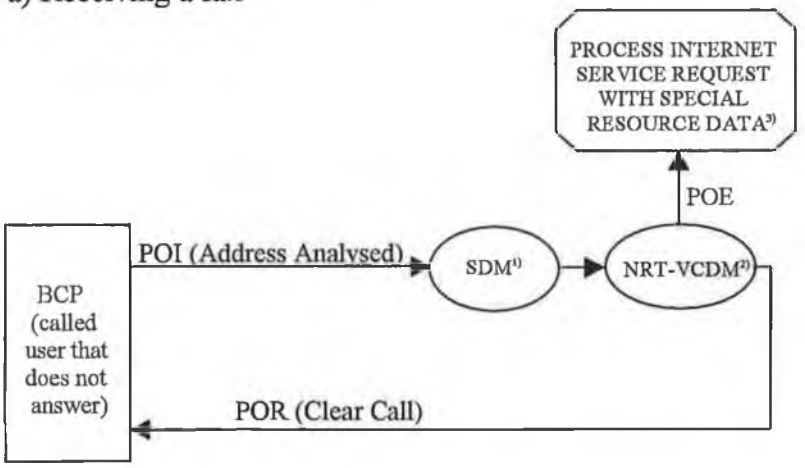


NOTES: 1) Collects information about the URL containing the document to be played as spoken content and the destination telephone number. Then sends a request to the 3pSRC to call the destination telephone number and play the content contained in the URL.

Figure C.16 – SIB & IASC Chains for the Request to Hear Content Service When Provided by a Third Party Special Resource



a) Receiving a fax



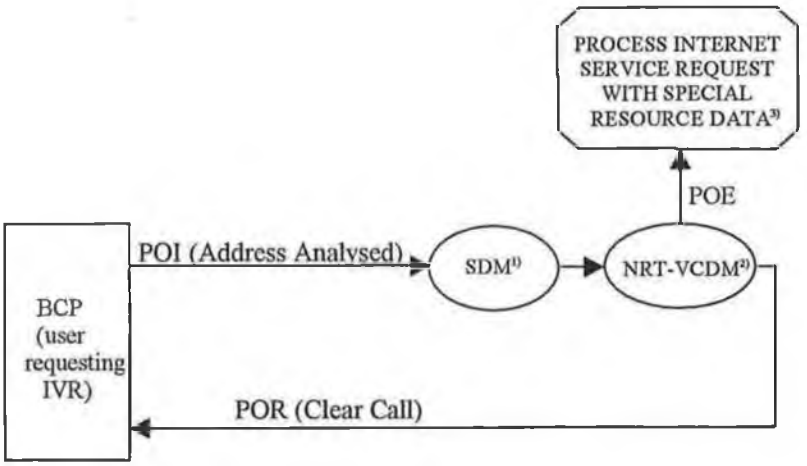
b) Receiving a voice message

NOTES: 1) Using the dialled number (which in the case of a fax is a "virtual" fax number and in the case of a voice message is the physical number of the called party), finds the e-mail address of the recipient

2) Instructs the Special Resource to receive fax or voice message and sends a request to the Internet to send the data, received by the Special Resource and converted to an Internet format in the Special Resource Gateway, to the specified address.

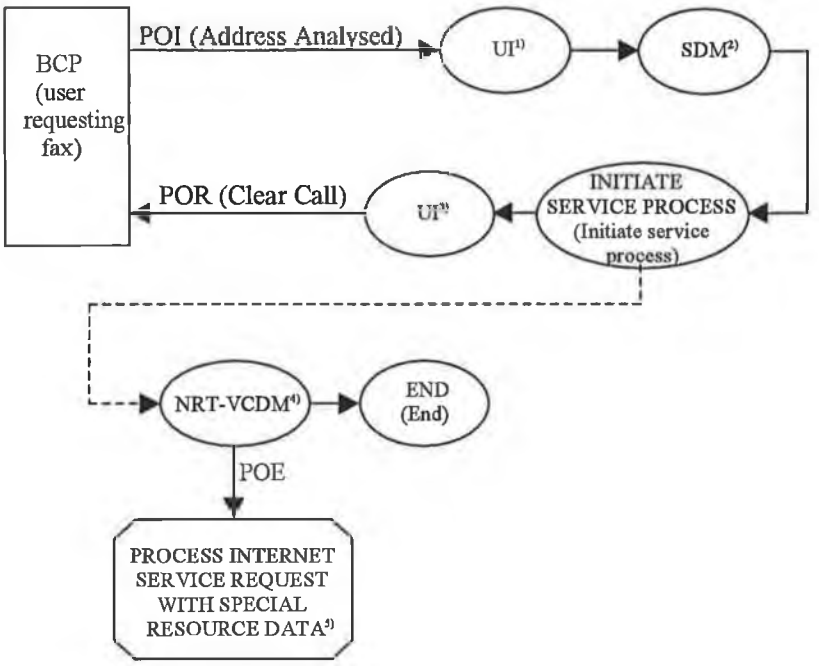
3) Sends the data received through the Special Resource Gateway to the specified e-mail address

Figure C.17 – SIB & IASC Chains for the Unified Messaging Service When Provided by an IN Special Resource



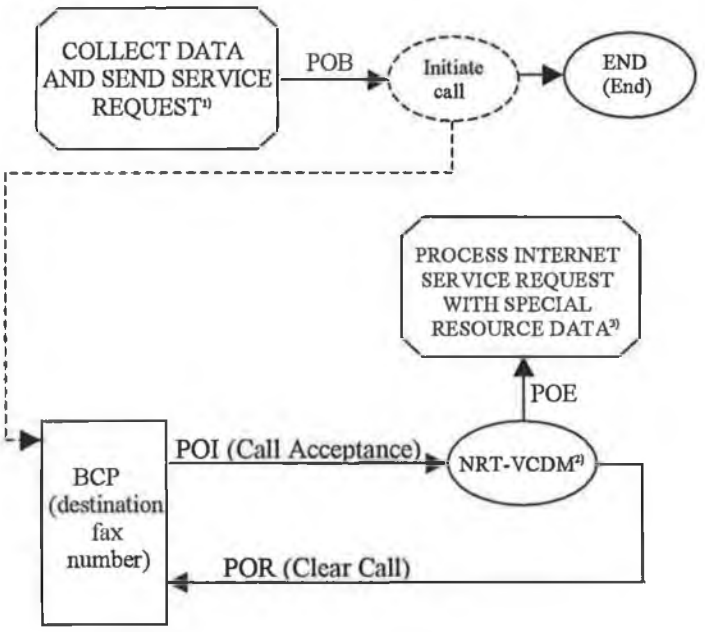
NOTES: 1) Retrieves the URL of the IVR script corresponding to the dialled number
2) Makes a request to the Internet part of the system for the IVR script with the URL got in 1) to be sent to the IN special resource and instructs the Special Resource to connect to the user requesting IVR and to execute the IVR script.
3) Fetches the IVR script from the specified URL and feeds it into the IN special resource gateway.

Figure C.18 – SIB & IASC Chains for the Interactive Voice Response (IVR) Service When Provided by the IN Special Resource



NOTES: 1) Performs digit collection – for destination fax number and document code
2) Determines the URL of the document to be faxed, based on the document code
3) Plays announcement to user to say that the fax has successfully been requested
4) Instructs the appropriate entity in the Internet to fetch the document to be faxed from the URL and to feed it into the special resource gateway for format conversion. Also instructs the special resource to fax the converted document to to the fax number acquired at digit collection.
5) Fetches the document at the specified URL and feeds it into the special resource gateway for format conversion.

Figure C.19 – SIB & IASC Chains for the Fax-on-Demand Service When Provided by an IN Special Resource

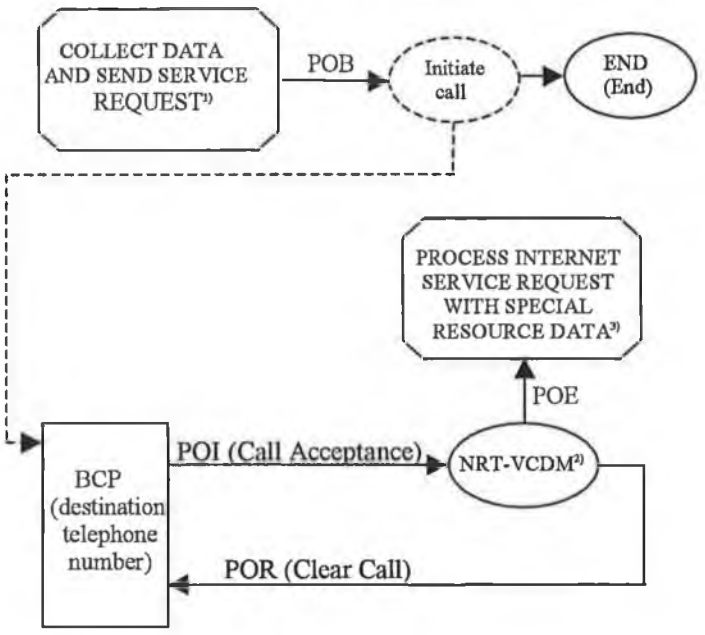


NOTES: 1) Collects information through an HTML form or Java applet about the URL of the document to be faxed and the destination fax number.

2) Instructs the appropriate entity in the Internet to fetch the document to be faxed from the URL and to feed it into the special resource gateway for format conversion. Also instructs the special resource to connect to the destination fax number and to fax the converted document to it.

3) Fetches the document at the specified URL and feeds it into the special resource gateway for format conversion.

Figure C.20 – SIB & IASC Chains for the Request-to-Fax Service When Provided by an IN Special Resource



NOTES: 1) Collects information through an HTML form or Java applet about the URL of the document containing the content to be played.

2) Instructs the appropriate entity in the Internet to fetch the document from the URL and to feed it into the special resource gateway for format conversion. Also instructs the special resource to connect to the destination telephone number and to send the converted document to it.

3) Fetches the document at the specified URL and feeds it into the special resource gateway for format conversion.

Figure C.21 – SIB & IASC Chains for the Request to Hear Content Service When Provided by an IN Special Resource

Appendix D

Inter-FE Messaging Formats for the IIIN

1. A simple text-based protocol

Following is a list of messages that might be used for communication between the Internet Functional Entities of the IIIN. The messages are in text format, similar to those of HTTP and SMTP. These messages are used for communication between the FEs in the IIIN prototype built as part of the M.Eng. project.

Notation:

- single words in capitals are keywords
- text within <> brackets describes a variable token, where a token is either a string of characters or a group of such strings between a pair of double quotes
- ... denotes that the previous line may be repeated 0 or more times.

Message from ISCGF to the IIIN Micro Server or the SPOISF

```
IIINREQUEST
COMPONENT <component code (integer)>
ADDRESS <e-mail address or URL (character string)>
PARAMETERS <number of parameters>
<parameter name> <value>
...
COMPLEX <parameter name>
<value line>
...
COMPLEX PARAMETER END
...
PARAMETERS END
IIINREQUEST END
```

Message from ISCGF to SRAIF

IIINREQUEST
TYPE <manipulation type code (integer)>
ADDRESS <e-mail address or URL (character string)>
IIINREQUEST END

Message from ISCGF to a 3pSRC

IIINREQUEST
ADDRESS <e-mail address or URL (character string)>
IIINREQUEST END

Messages to ISCGF from other FEs

IIINREQUEST
COMPONENT <component code (integer)>
PARAMETERS <number of parameters>
<parameter name> <value>
...
COMPLEX <parameter name>
<value line>
...
COMPLEX PARAMETER END
...
PARAMETERS END
IIINREQUEST END

2. IDL Interfaces

Following is the listing for the Interface Definition Language (IDL) module that contains the interfaces on all the IIIN FEs receiving requests in the Internet. These interfaces have been used to compile and implement CORBA objects that represent those FEs in the IIIN prototype built as part of this M.Eng. project

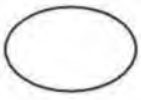
```
// This idl module contains all the interfaces for CORBA inter-FE
// communication in the IIIN


module IIIN
{
   // interface to the IIIN Micro Server
   interface IIINMicroServer {

        void ExecuteServiceComponent
        (in unsigned short sc_type,
         in string relevant_address,
                in any parameters);
   };

   // interface to the Service Provider Owned Internet
   // Server Function
   interface ServiceProviderOwnedInternetServer {

        void ExecuteServiceComponent
        (in unsigned short sc_type,
                in string relevant_address,
                in any parameters);
   };

   // interface to the Special Resource Assisting Internet Function
   interface SpecalResourceAssistant {

        void ManipulateSpecialResourceData
        (in unsigned short manip_type,
                in string relevant_address);
   };

   // interface to the Third Party Special Resource Controller
   interface ThirdPartySpecialResourceController {

        void Action (in string relevant_address);
   };

   // interface to the Internet-Service Control Function Gateway
   interface InternetSCFGateway {

        void ExecuteServiceComponent
        (in unsigned short component_type,
                in any parameters);
   };
};
```

Appendix E

SIB Chains Representing the Sample IN/IIIN Services Implemented to Demonstrate the IN/IIIN Platform Prototype Functionality

Two sample IN services have been implemented: Abbreviated Dialling and Call Forwarding (unconditional). SIB chains of the implementations of both of those services are presented here. As for the IIIN sample services, four have been implemented: Customer Profile Management (for the registration of an IIIN user), Click-to-Dial-Back, Group Announcements and Parallel Routing. The SIB chains for the implementations of the three last services are identical to those given in Appendix C. The implementation SIB chains of IIIN User Registration (as an example of CPM) are shown in this Appendix.

The following graphical symbols are used:



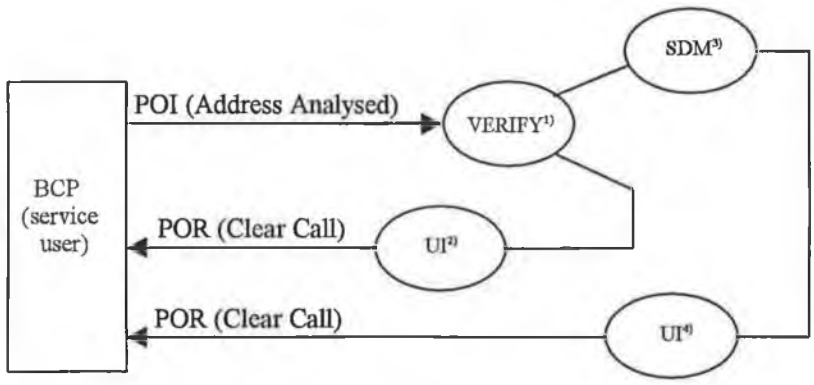
SIB

Basic Call Process, with a description of user at the side at which the service logic initiating Detection Point is

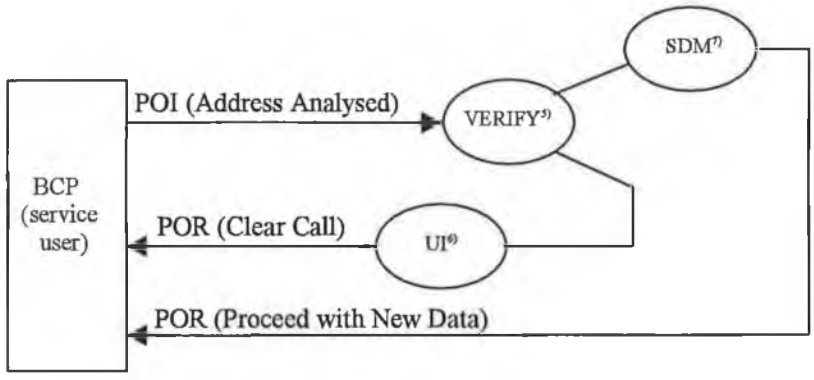
Service logic flow

The following acronyms are used:

BCP	Basic Call Process
POI	Point of Initiation
POR	Point of Return
UI	User Interaction
SDM	Service Data Management



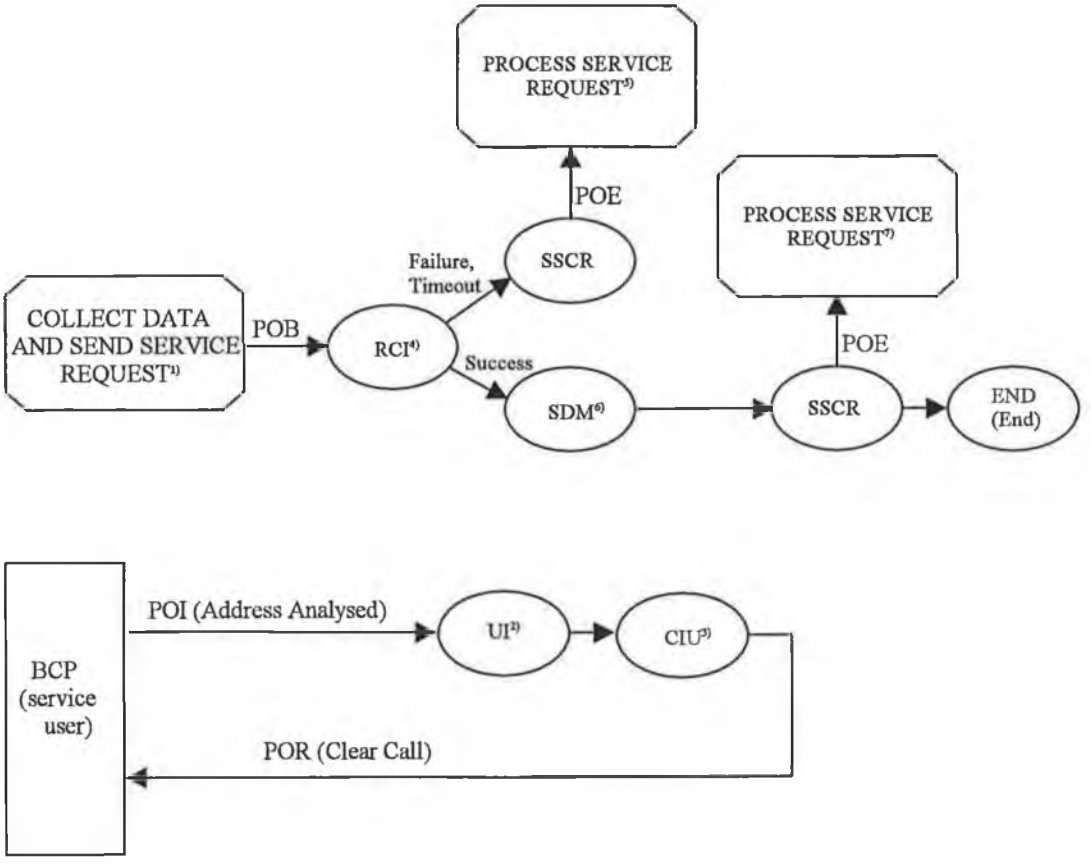
a) Service activation and deactivation



b) Service operation

NOTES: 1) Verifies the validity of the string for service activation/deactivation
2) Plays announcement to user informing him/her that the dialled string is invalid
3) Accesses the database to add/remove user specific data for the service in question
4) Plays announcement to user confirming successful activation/deactivation
5) Verifies service request string
6) Plays announcement to user informing him/her that the dialled string is invalid
7) Accesses user specific data for the particular service (in the case of Abbreviated dialling it is the destination telephone number corresponding to the abbreviated sequence that was dialled; in the case of Call Forwarding (unconditional) it is the destination number stored in the "unconditional forwarding number" field for the dialled number)

Figure E.1 – SIB Chains for the Abbreviated Dialling and Call Forwarding IN services, as Implemented for the Purpose of Demonstrating the Functionality of the Basic IN Platform Prototype



NOTES: 1) Receives a request for IIIN registration from a user; generates a random sequence of digits; sends this random sequence, together with the data gathered from the user, in a request for service to the IN; asks the user to ring a certain number and enter the digit sequence when prompted in order to complete the registration.
2) Collects a sequence of digits from the user
3) Posts the collected digits as Call Information, for the other SIB chain to retrieve.
4) Retrieves the Call Information, specifying the required code as the "Value to Match" CID Parameter
5) Pops an error message onto the user's screen, since the confirmation was not done within the given time or the code entered through the telephone was invalid
6) Adds the newly-registered user's data into the database (this may take more than one call to the SDM SIB)
7) Pops a registration acknowledgement onto the user's screen

Figure E.2 – SIB & IASC Chain for the Customer Profile Management Service Feature Applied to IIIN User Registration, as Implemented for the Purpose of Demonstrating the IIIN Prototype Functionality

Appendix F

Detailed UML Descriptions of the Classes Implementing the Basic IN and the IN Part of the IIIN

Following are detailed UML descriptions of the classes that implement the basic IN and the IN part of the IIIN. They contain all the attributes and the externally visible (public) operations for the classes.

The most drastic changes to the IN system, as it was extended to become the IIIN, were made in the FRL class of the SCF, and this has been noted in the UML description of that class. The other classes have been changed slightly or only in the implementation of the operations.

Apart from the classes appearing in the high-level class diagrams in Chapter 5 (§5.3.1.2.1 and §5.3.1.2.2) there are some other classes described here that fall into two categories. The first category are "message classes", the role of which is to hold the messages passing between classes of the SSF at different stages of processing. The message classes are switch_msg_form, ccaf_msg_form and scfam_msg_form. In the implementation, there is one globally declared instance of each type of message, which is enough considering that only one thread in the process has access to these messages.

The second category of classes appearing here and not in the high-level class diagrams are those defined within classes visible on the highest level. Such classes are contained in the FRL class and they are IIIN, CS_1 and CS_2. Further, each of these three classes contain classes which implement the SIBs needed for the IIIN sample services, and these are also represented with separate diagrams.

Objects of the ssf Module

UML	**switch_msg_form**
	-msg_status:Octet -MessageStruct:StructureDefinition -rcv_msg:MessageStruct -rcv_msg:MessageStruct
	+switch_msg_form () +~switch_msg_form () +initialise (data:OctetPointer) +clear ()
Purpose	Stores a switch message after receiving raw data from the switch or immediately before it is sent to the switch. The structure is processed by the CCAF.
Description of attributes and operations	**msg_status:Octet** : Switch-specific message status. **MessageStruct:StructureDefinition** : Definition of structure to hold the received and sent switch message. **rcv_msg:MessageStruct** : Received message. **snd_msg:MessageStruct** : Message to be sent.
	switch_msg_form () : Constructor. **~switch_msg_form ()** : Destructor. **initialise (data:OctetPointer)** : Initialises the rcv_msg from raw data. **clear ()** : Clears structures for use with new messages.
NOTES	The function CCAF::process_switch_message () is a friend to this class i.e. has access to the invisible attributes.

UML	**ccaf_msg_form**
	-line:Octet **-event:Octet** **-pdata:CharacterPointer** **-data_size:Integer**
	+ccaf_msg_form () **+ccaf_msg_form (otherObj:ccaf_msg_formPointer)** **+~ccaf_msg_form();** **+initialise(UBYTE,UBYTE,UBYTE *, int)** **+clear()** **+rtn_line ():Octet** **+rtn_event ():Octet** **+rtn_pdata ():CharacterPointer** **+rtn_data_size ():Integer**
Purpose	Stores a message from the CCAF for the BCM.
Description of attributes and operations	**line:Octet** : Line number associated with the message. **event:Octet** : Message type (event). **pdata:CharacterPointer** : Any data that may be relevant to the message type, e.g. dialled digits. **data_size:Integer** : Size of the data.
	ccaf_msg_form () : Constructor. **ccaf_msg_form (ccaf_msg_form &)** : Copy constructor. **~ccaf_msg_form()** : Destructor. **initialise (lineNo:Octet, switchEvent:Octet, data:CharacterPointer, dataSize:Integer)** : Initialises the message with the values given as parameters. **clear()** : Clears the structure to hold the next message. **rtn_line ():Octet** : Returns the value of the line attribute. **rtn_event ():Octet** : Returns the value of the event attribute. **rtn_pdata ():CharacterPointer** : Returns the value of the pdata attribute. **rtn_data_size ():Integer** : Returns the value of the data_size attribute.

UML	**scfam_msg_form**
	-last:Integer **-next:scfam_msg_formPointer** **-action:Octet** **-direction:Octet** **-line_no:Integer** **-pData:CharacterPointer**
	+scfam_msg_form () **+scfam_msg_form (otherObj:scfam_msg_formPointer)** **+~scfam_msg_form ()** **+copy_and_clear ()** **+clear()**
Purpose	Stores a message used for communication between the SCFAM and the BCM/BCSM.
Description of attributes and operations	**last:Integer** : Indicates if the current message is the last one already received from the SCF. **next:scfam_msg_formPointer** : If last is 0, points to the next message received from the SCF. **action:Octet** : Indicates the action to be performed on the call. **direction:Octet** : Indictes if the message is a request or a response. **line_no:Integer** : The line number associated with the message. **pData:CharacterPointer** : Any data associated with the message.
	scfam_msg_form () : Constructor. **scfam_msg_form (otherObj:scfam_msg_formPointer)** : Copy constructor. **~scfam_msg_form ()** : Destructor. **copy_and_clear ()** : Clears the current message and makes the next message current. **clear()** : Clears the message.
NOTES	The function SCFAM and BCSM classes are friends to this class i.e. they have access to the invisible attributes.

UML	**CCAF**
	-line_table:OctetMatrix **-digitCounter:IntegerArray** **-numDigits:IntegerArray** **-digitBuffer:OctetMatrix** **-collect_again:BooleanArray**
	+CCAF () **+~CCAF ()** **+init_line_table ()** **+process_switch_msg ()** **+play_tone (toneType:Integer, lineNum:Integer)** **+disconnect_tone (lineNum:Integer)** **+start_digit_collection (lineNum:Integer)** **+connect (origLineNum:Integer, lineNum:Integer)**
Purpose	Implements the CCAF Functional Entity of the IN as well as some of the CCF FE functionality.
Description of attributes and operations	**line_table:OctetMatrix** : Table of lines against physical channels. **digitCounter:IntegerArray** : Array of counters for received digits on each line. **numDigits:IntegerArray** : Array of number of expected digits for each line. **digitBuffer:OctetMatrix** : Digit buffer for each line. **collect_again:BooleanArray** : Indicator used by digit collection logic.
	CCAF () : Constructor. **~CCAF ()** : Destructor. **init_line_table ()** : Initialises the table of lines against assigned telephone numbers; **process_switch_msg ()** : Processes a message received from the switch. **play_tone (toneType:Integer, lineNum:Integer)** : Instructs the switch to play the tone of type toneType to line lineNum. **disconnect_tone (lineNum:Integer)** : Instructs the switch to deactivate the tone being played to line lineNum. **start_digit_collection (lineNum:Integer)** : Instructs the switch to start collecting digits from line lineNum. **connect (origLineNum:Integer, lineNum:Integer)** : Instructs the switch to connect line origLineNum to line lineNum.

UML	**BCM**
	-DN_table:CharacterMatrix **-bcsm_table:BCSMPointerArray** **-dp_mngr_table:DPManagerPointerArray**
	+BCM (); **+~BCM ();** **+process_ccaf_msg ()** **+process_scfam_msg ()** **+process_scfam_msg_noline ()** **+process_timer_expired (int lineNum)**
Purpose	Implements the Basic Call Manager, which is a part of the CCF Functional Entity of the IN.
Description of attributes and operations	**DN_table:CharacterMatrix** : Table of dialling numbers against line numbers. **bcsm_table:BCSMPointerArray** : Array of pointers to active BCSM instances. **dp_mngr_table:DPManagerPointerArray** : Array of pointers to DPManagers associated with the active BCSM instances.
	BCM () : Constructor. **~BCM ()** : Destructor. **process_ccaf_msg ()** : Processes a message received from the CCAF. **process_scfam_msg ()** : Processes a message from the SCF. **process_scfam_msg_noline ()** : Processes a message from the SCF that is not associated with a call. Called from process_scfam_msg (). **process_timer_expired (int lineNum)** : Handles the expiry of a timer for line lineNum.

UML	**BCSM**
	-state:Integer **-line_no:Integer** **-bcsmType:BCSMType** **-pdigits:CharacterArray** **-exception_parameter:Integer** **-currentDP:Integer** **-currentCriterion:Integer** **-currentTDPtype:TDP_TYPE** **-dp_state:Integer** **-SavedEvents:SavedEventStructArray** **-SvdEvent:Integer** **-uIdTimerEvent:UnsignedInteger** **-dpManID:Integer** **-msgErrno:Integer** **-forwardedList:CharacterPointerArray** **-TDP criteria met:FunctionPointerArray**
	+BCSM (lineNo:Integer, bcsmType:BCSMType); **+~BCSM ()** **+event_occurred(evt:Integer):RTNCODE** **+process_scfam_msg_withline():RTNCODE**
Purpose	Virtual base class for the originating and terminating types of the Basic Call State Model (see objects OBCSM and TBCSM).
Description of attributes and operations	**state:Integer** : BCSM state, such as O_NULL_AUTH_ORIG_ATT. **line_no:Integer** : The line number that this BCSM is associated with. **bcsmType:BCSMType** : Type of BCSM (Originating or Terminating). **pdigits:CharacterArray** : Received digits. **exception_parameter:Integer** : Indicates the type of exceptional circumstances, if such occur. **currentDP:Integer** : The value of the DP currently being processed. **currentCriterion:Integer** : The current criterion being processed for the current DP. **currentTDPtype:TDP_TYPE** : The type of the current DP if it is a TDP. **dp_state:Integer** : Indicator as to whether the BCSM has been suspended at the current DP. **SavedEvents:SavedEventStructArray** : An array storing the events that occurred during the processing of the current DP. **SvdEvent:Integer** : Value of the saved event currently being processed. **uIdTimerEvent:UnsignedInteger** : Timer event id variable. **dpManID:Integer** : ID of the DPManager instance associated with this BCSM. **msgErrno:Integer** : Holds the error number when an error is encountered during SCF message processing. **forwardedList:CharacterPointerArray** : List of forwarded numbers, as received from the SCF, for TDP criteria processing. **TDP criteria met:FunctionPointerArray** : Contains pointers to TDP criteria processing functions.
	BCSM (lineNo:Integer, bcsmType:BCSMType) : Constructor. **~BCSM ()** : Destructor. **event_occurred(evt:Integer):RTNCODE** : Is overridden in the O_BCSM and T_BCSM classes, processes event evt from the CCAF. **process_scfam_msg_withline():RTNCODE** : Process a message from the SCF associated with a particular call. Called from BCM::process_scfam_msg ( ).

UML	**OBCSM**
	-term_line_no:Integer **-ptemp:TBCSMPointer**
	+OBCSM (lineNo:Integer) **+~OBCSM ()** **+event_occurred (evt:Integer):RTNCODE** **+scf_initiated ()**
Purpose	Implements the Originating Basic Call State Model, which describes the originating-side call processing that takes place in the CCF/SSF functional entity of the IN.
Description of attributes and operations	**term_line_no:Integer** : Line number with which the corresponding TBCSM is associated. **ptemp:TBCSMPointer** : Pointer to the corresponding TBCSM.
	OBCSM (lineNo:Integer) : Constructor. **~OBCSM ()** : Destructor. **event_occurred (evt:Integer):RTNCODE** : Overridden from BCSM, processes event evt from the CCAF. **scf_initiated ()** : Marks the OBCSM as one initiated from the SCF, rather than from the CCAF.

UML	**TBCSM**
	-orig_line_no:Integer
	+TBCSM (lineNo:Integer) **+~TBCSM ()** **+event_occurred (evt:Integer):RTNCODE**
Purpose	Implements the Terminating Basic Call State Model, which describes the terminating-side call processing that takes place in the CCF/SSF functional entity of the IN.
Description of attributes and operations	**orig_line_no:Integer** : Line number with which the corresponding OBCSM is associated.
	TBCSM (lineNo:Integer) : Constructor. **~TBCSM ()** : Destructor. **event_occurred (evt:Integer):RTNCODE** : Overridden from BCSM, processes event evt from the CCAF.

UML	**DPManager**
	-tdpArmflags:IntegerArray **-edpArmflags:IntegerArray** **-currentCriteria:Integer** **-relationship:Integer**
	+DPManager () **+~DPManager ()** **+initialise tdpArmflags ()** **+EDP_armed (edp:Integer):EDP_TYPE** **+TDP_armed (tdp:Integer)TDP_TYPE** **+arm(edp:Integer, edpType:EDP_TYPE):RTNCODE** **+disarm (edp:Integer):RTNCODE** **+existing_relationship ():Integer** **+open_relationship():RTNCODE** **+new_relationship(currentDP:Integer):Integer**
Purpose	Implements DP management functionality which assists the BCSM DP processing by keeping track of the arming and disarming of DPs.
Description of attributes and operations	**tdpArmflags:IntegerArray** : Contains flags indicating, for each DP, if it is armed as a TDP. **edpArmflags:IntegerArray** : Contains flags indicating, for each DP, if it is armed as an EDP. **currentCriteria:Integer** : Current criteria being processed. **relationship:Integer** : Indicates if a relationship currently exists and what type it is.
	DPManager () : Constructor. **~DPManager ()** : Destructor. **initialise tdpArmflags ()** : Arms the TDPs according to static data read from a file. **EDP_armed (edp:Integer):EDP_TYPE** : Checks if EDP edp is armed and returns the EDP type if it is. **TDP_armed (tdp:Integer)TDP_TYPE** : Checks if TDP tdp is armed and returns the TDP type if its is. **arm(edp:Integer, edpType:EDP_TYPE):RTNCODE** : Dynamically arms EDP edp as type edpType. **disarm (edp:Integer):RTNCODE** : Dynamically disarms EDP edp. **existing_relationship ():Integer** : Returns the relationship ID or NULL_RELATIONSHIP if there is no relationship. **open_relationship():RTNCODE** : Opens a control relationship between the SSF and SCF for the current call. **new_relationship(currentDP:Integer):Integer** : Opens or changes type of relationship based on the armed EDPs for the current call.

UML	**SCFAM**
	-scfRcvMsg:SCFReceivedMessageStruct **-callIDTable:OctetMatrix** **-index:Integer** **-messageBuffer:OctetArray** **-scfMsgQList:SCFMessageQueueStruct**
	+SCFAM () **+~SCFAM ()** **+send_TDP_msg (lineNo:Integer, bcsmType:BCSMType, tdpName:Integer, tdpType:TDP_TYPE, criterionData:CharacterPointer):RTNCODE** **+send_EDP_msg (lineNo:Integer, bcsmType:BCSMType, edpName:Integer, edpType:EDP_TYPE, data:CharacterPointer):RTNCODE** **+rcv_scf_msg (data:OctetPointer)** **+process_scf_msg ():RTNCODE** **+process_message_for_scf ():RTNCODE** **+delete_relationship (lineNo:Integer, bcsmType:BCSMType)**
Purpose	Implements the SCF Access Manager component of the SSF IN Functional Entity.
Description of attributes and operations	**scfRcvMsg:SCFReceivedMessageStruct** : Structure to hold the current SCF message when the raw data is unpacked. **callIDTable:OctetMatrix** : Holds various call information. **index:Integer** : Index in the call id table of the call currently being processed. **messageBuffer:OctetArray** : Buffer for outgoing messages. **scfMsgQList:SCFMessageQueueStruct** : Holds the message queues for all current calls.
	SCFAM () : Constructor. **~SCFAM ()** : Destructor. **send_TDP_msg (lineNo:Integer, bcsmType:BCSMType, dpName:Integer, tdpType:TDP_TYPE, criterionData:CharacterPointer):RTNCODE** : Builds a message for the SCF at TDP tdpName of type tdpType, for line lineNo associated with a bcsm of type bcsmType. The message is built using the criterionData, provided by the BCSM at DP processing. **+send_EDP_msg (lineNo:Integer, bcsmType:BCSMType, edpName:Integer, edpType:EDP_TYPE, data:CharacterPointer):RTNCODE** : Builds a message for the SCF at EDP edpName of type edpType, for line lineNo associated with a bcsm of type bcsmType. The message is built using the ata, provided by the BCSM at DP processing. **rcv_scf_msg (data:OctetPointer)** : Extracts raw data received from the SCF into a structure. **process_scf_msg ():RTNCODE** : Processes the received scf msg. **process_message_for_scf ():RTNCODE** : Processes messages sent by send_TDP_msg or send_EDP_msg. The main role of this function is to intercept the sending of messages that already have answers received, in the queue. This is especially important in the case of SCF initiated calls, for which several messages are received together. **delete_relationship (lineNo:Integer, bcsmType:BCSMType)** : Deletes relationship with SCF for line lineNo with bcsm of type bcsmType.

UML	**PipeReaderWriter**
	-hHSIPipe:Handle **-hSCFPipe:Handle**
	+PipeReaderWriter () **+~PipeReaderWriter ()** **+initialise ():RTNCODE** **+connect_to_HSI ():RTNCODE** **+connect_to_scf ():RTNCODE** **+write_pipe (pipeId:Integer, write_buf:OctetPointer, buf_len:Integer):RTNCODE** **+read_pipe (pipeId:Integer, read_buf:OctetPointer, buf_len:Integer):RTNCODE**
Purpose	Performs the low-level communication of the CCF/SSF with the on the pipes towards the switch and the SCF.
Description of attributes and operations	**hHSIPipe:Handle** : Handle to host-switch interface pipe. **hSCFPipe:Handle** : Handle to SCF pipe.
	PipeReaderWriter () : Constructor. **~PipeReaderWriter ()** : Destructor. **initialise ():RTNCODE** : Creates and initialises the pipes. **connect_to_HSI ():RTNCODE** : Connect to host-switch interface pipe. **connect_to_scf ():RTNCODE** : Connect to pipe for communication with the SCF. **write_pipe (pipeId:Integer, write_buf:OctetPointer, buf_len:Integer):RTNCODE** : Write the data in write_buf, with length buf_len, to pipe with handle hPipId. **read_pipe (pipeId:Integer, read_buf:OctetPointer, buf_len:Integer):RTNCODE** : Read data from pipe with handle hPipeId into read_buf, with length buf_len.

Objects of the scf Module

UML	**FEAM**
	-hSDFBufSemaphore:Handle -hISCGFPipe:Handle -hSSFPipe:Handle -hSDFBufSemaphore:Handle -pEventHandles:HandleArray -dwEvent:Integer -hISCGFWriteMutex:Handle -hSSFWriteMutex:Handle -hSDFWriteMutex:Handle -hSRFWriteMutex:Handle -threadHandleList:HandleList -hThreadHandleListMutex:Handle -pIflowReadBuf:OctetArray -pIflow:OctetArray -pISCGFiflowReadBuf:OctetArray -pISCGFiflow:OctetArray -pSDFiflowReadBuf:OctetArray -pSDFiflow:OctetArray -onlyFEAM:FEAMPointer
	+FEAM () +~FEAM (); +wait_for_events() +connectSDF_SCF (messageBuffer:OctetPointer, hSemaphore:Handle):Handle +add thread handle (newHandle:Handle) +send message (fe:FunctionalEntity, msg:OctetPointer, len:Integer):Integer
Purpose	Implements the FEAM component of the SCF FE of the IN.

Description of attributes and operations	**hSDFBufSemaphore:Handle** : Handle to semaphore for communication with SDF. **hISCGFPipe:Handle** : Handle to pipe towards the ISCGF. **hSSFPipe:Handle** : Handle to pipe towards the SSF. **hSDFBufSemaphore:Handle** : Handle to semaphore for communication with SDF. **pEventHandles:HandleArray** : Array of handles to all the possible read/write and thread exit events. **dwEvent:Integer** : Last read/write event that occurred. **hISCGFWriteMutex:Handle, hSSFWriteMutex:Handle, hSDFWriteMutex:Handle, hSRFWriteMutex:Handle** : Mutices for synchronising reading/writing to the pipes by the SLPs. **threadHandleList:HandleList** : List of handles to the currently running SLP threads, used to signal SLP exits for the purpose of cleanup. **hThreadHandleListMutex:Handle** : Mutex for synchronising access to the thread handle list. **pIflowReadBuf:OctetArray** : Buffer for receiving data from SSF. **pIflow:OctetArray** : Buffer for the processing of SSF incoming data. **pISCGFiflowReadBuf:OctetArray** : Buffer for receiving data from ISCGF. **pISCGFiflow:OctetArray** : Buffer for the processing of ISCGF incoming data. **pSDFiflowReadBuf:OctetArray** : Buffer for receiving data from SDF. **pSDFiflow:OctetArray** : Buffer for the processing of SDF incoming data. **onlyFEAM:FEAMPointer** : Pointer to the only instance of the FEAM in the system.
	FEAM () : Constructor. **~FEAM ();** : Destructor. **wait_for_events()** : Waits for events from ths SSF and from the ISCGF. **connectSDF_SCF (messageBuffer:OctetPointer, hSemaphore:Handle):Handle** : Connects to the SDF to communicate with it within the process through the buffer messageBuffer and the semaphore with handle hSemaphore. **add thread handle (newHandle:Handle)** : Adds a handle to an SLP thread to the wait list, so that cleanup can be made once the SLP exits. **send message (fe:FunctionalEntity, msg:OctetPointer, len:Integer):Integer** : Sends message msg, of length len, to Functional Entity fe.

UML	**SLEM**
	-callID:IntegerArray **-hMsgRcvdEvent:HandleArray** **-pInstDataTable:SlpInstDataStructPointer** **-pSdfDataTable:SdfDataStructPointer** **-hSLPthread:HandleArray** **-threadID:IntegerArray** **-onlySLEM:SLEMPointer**
	+SLEM () **+~SLEM ()** **+processIflow (data:OctetArray)** **+processISCGFiflow (data:OctetArray)** **+processSDFiflow (data:OctetArray)** **+get slp value (value type:Integer):VoidPointer** **+slp thread exited (hnd:Handle)**
Purpose	Implements the SLEM component of the SCF FE of the IN.
Description of attributes and operations	**callID:IntegerArray** : Array of currently relevant call ids. **hMsgRcvdEvent:HandleArray** : Array of handles (with indices matching those of corresponding call ids in the callID array) to events used to signal to SLPs that a message has arrived from another FE. **pInstDataTable:SlpInstDataStructPointer** : rray of pointers (with indices matching those of corresponding call ids in the callID array) to a structure for receiving data from the SSF, for each SLP. **pSdfDataTable:SdfDataStructPointer** : Array of pointers (with indices matching those of corresponding call ids in the callID array) to a structure for receiving data from the SDF, for each SLP. **hSLPthread:HandleArray** : Array of handles (with indices matching those of corresponding call ids in the callID array) to the SLP threads. **threadID:IntegerArray** : Array of thread ids (with indices matching those of corresponding call ids in the callID array) of the SLP threads. **onlySLEM:SLEMPointer** : Pointer to the only instance of the SLEM in the system.
	SLEM () : Constructor. **~SLEM ()** : Destructor. **processIflow (data:OctetArray)** : Processes a message received from the SSF. **processISCGFiflow (data:OctetArray)** : Processes a message received from the ISCGF **processSDFiflow (data:OctetArray)** : Processes a message received from the SDF. **get slp value (value type:Integer):VoidPointer** : Gets a value associated with the calling SLP of type value_type. The value may be the relationship id, the call id or received data. **slp thread exited (hnd:Handle)** : Does the cleanup for an SLP thread with handle hnd.

UML	FRL
	-ServiceFeatureEntryStruct:StructureDefinition -serviceFeatureTable:ServiceFeatureEntryStructArray *(Additional attributes for the IIIN)* -onlyFRL:FRLPointer -IIINSFTElement:StructureDefinition -iiinSFT:IIINSFTElementList -IIINDBRecord:StructureDefinition -iiinDB:IIINDBRecordList +SystemException:ClassDefinition +IIIN:ClassDefinition +CS 1:ClassDefinition +CS 2:ClassDefinition
	+FRL () +~FRL () +SDM (ssd:SDMSSDStrucPointer, info:SlpInfoStructPointer):Integer +Translate (ssd:TRANSLATESSDStructPointer, info:SlpInfoStructPointer):Integer +Verify (ssd:VERIFYSSDStructPointer):Integer +DetermineServiceFeature (if:IFLOWNAME, data:OctetPointer): LPTHREAD_START_ROUTINE *(Additional operations for the IIIN)* +send ssf message (message type:Integer, relationship id:Integer, call id:Integer, data1 p:VoidPointer, data2 p:VoidPointer):Integer +send iscgf message (msg:IscgfScfMessageDataStructPointer):Integer +extract iiin slp data (dataList:StringList, indata p:OctetPointer, indata len:Integer) +call SDM (ssd:SDMSSDStructPointer):Integer +get relationship id ():Octet +get call id ():Octet +get event handle ():Handle +get in data ():OctetPointer +get in data len ():Integer +get in type ():IFLOWNAME +get iiin service feature (serviceKey:Octet): LPTHREAD START ROUTINE
Purpose	Implements the FRL component of the SCF FE of the IN.

Description of attributes and operations	**ServiceFeatureEntryStruct:StructureDefinition** : Structure that holds a service key and an address to the associated SLP routine. **serviceFeatureTable:ServiceFeatureEntryStructArray** : Array used for the identification of the SLP to be run, when a request for a service feature is received from the SSF. ***(Additional attributes for the IIIN)*** **onlyFRL:FRLPointer** : Pointer to the only instance of FRL. **IIINSFTElement:StructureDefinition** : Structure that associates a service key with an SLP-implementing function. **iiinSFT:IIINSFTElementList** : List of structures associating the appropriate service keys with the available SLP-implementing functions for the IIIN. **IIINDBRecord:StructureDefinition** : IIIN database structure definition. **iiinDB:IIINDBRecordList** : List of IIIN database records. Used by the SDF as the database, in the case of resolving IIIN related queries. **SystemException:ClassDefinition** : Exception class for SIB exceptions. **IIIN:ClassDefinition** : Defines the class that contains the IIIN-specific SIBs. **CS 1:ClassDefinition** : Defines the class that contains some CS-1 SIBs required by the IIIN. **CS 2:ClassDefinition** : Defines the class that contains CS-2 SIBs required by the IIIN.

	FRL () : Constructor. **~FRL ()** : Destructor. **SDM (ssd:SDMSSDStrucPointer, info:SlpInfoStructPointer):Integer** : Implements the SDM SIB. **Translate (ssd:TRANSLATESSDStructPointer, info:SlpInfoStructPointer):Integer** : Implements the TRANSLATE SIB. **Verify (ssd:VERIFYSSDStructPointer):Integer** : Implements the VERIFY SIB. **DetermineServiceFeature (if:IFLOWNAME, data:OctetPointer): LPTHREAD_START_ROUTINE** : Determine which SLP should be started, based on the information flow type and data received. ***(Additional operations for the IIIN)*** **send ssf message (message type:Integer, relationship id:Integer, call id:Integer, data1 p:VoidPointer, data2 p:VoidPointer):Integer** : Sends a message to the SSF. This function is for use by SIBs and SLPs. **send iscgf message (msg:IscgfScfMessageDataStructPointer):Integer** : Sends a message to the ISCGF. This function is for use by SIBs and SLPs. **extract iiin slp data (dataList:StringList, indata p:OctetPointer, indata len:Integer)** : Accessory function used by SLPs and SIBs for the extraction of data from a message received from the ISCGF. **call SDM (ssd:SDMSSDStructPointer):Integer** : A function that adjusts the SDM SIB call to the new IIIN SLP format introduced with the IIIN-related extension of the system. **get relationship id ():Octet, get call id ():Octet, get event handle ():Handle, get in data ():OctetPointer, get in data len ():Integer, get in type ():IFLOWNAME** : Functions that allow the SIBs and SLPs to access various data related with the service instance. **get iiin service feature (serviceKey:Octet): LPTHREAD START ROUTINE** : Returns the SLP associated with a service key, for SLPs initiated from the ISCGF.

UML	**IIIN** +DynamicCallInfo:ClassDefinition +CIU:ClassDefinition +NRT VCDM:ClassDefinition +RCI:ClassDefinition +SSCR:ClassDefinition +TIMER:ClassDefinition +CALL LIST:ClassDefinition
Purpose	Contains the classes implementing the IIIN SIBs.
Description of attributes and operations	**DynamicCallInfo:ClassDefinition** : Defines a class that holds one Dynamic Call Information item (used by the CIU and RCI SIBs). **CIU:ClassDefinition** : Defines class that implements the CIU SIB. **NRT VCDM:ClassDefinition** : Defines class that implements the NRT-VCDM SIB. **RCI:ClassDefinition** : Defines class that implements the RCI SIB. **SSCR:ClassDefinition** : Defines class that implements the SSCR SIB. **TIMER:ClassDefinition** : Defines class that implements the TIMER SIB. **CALL LIST:ClassDefinition** : Defines class that implements the CALL LIST SDB.

UML	**CS_1** +USER INTERACTION:ClassDefinition
Purpose	Contains the classes implementing some CS_1WS SIBs needed by the IIIN.
Description of attributes and operations	**USER INTERACTION:ClassDefinition** : Defines class that implements the USER INTERACTION SIB.

UML	**CS_2** +END:ClassDefinition
Purpose	Contains the classes implementing the CS_2 SIBs needed by the IIIN.
Description of attributes and operations	**END:ClassDefinition** : Defines class that implements the END SIB.

UML	**CIU**
	+O_SUCCESS:IntegerConstant **+O_ERROR:IntegerConstant** **+ERR_INVALID_ID:IntegerConstant** **+ERR_INVALID_INFO:IntegerConstant** **+ERR_ID_NOT_FOUND:IntegerConstant** **+CIUSSD:StructureDefinition** **+ciuSsd:CIUSSD**
	+CIU () **~CIU ()** **+add_ci ():Integer** **+remove_ci ():Integer**
Purpose	Implements the CIU SIB of the IIIN.
Description of attributes and operations	**O_SUCCESS:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_INVALID_ID:IntegerConstant,** **ERR_INVALID_INFO:IntegerConstant,** **ERR_ID_NOT_FOUND:IntegerConstant** : Error value constants. **CIUSSD:StructureDefinition** : CIU SIB Service Specific Data structure. **ciuSsd:CIUSSD** : Holds the Service Specific Data.
	CIU () : Constructor. **CIU ()** : Desctructor. **add_ci ():Integer** : Posts a piece of dynamic call information. **remove_ci ():Integer** : Removes a piece of dynamic call information.

UML	**NRT_VCDM**
	+O_SUCCESS:IntegerConstant **+O_ERROR:IntegerConstant** **+ERR_INVALID_INTERNET_ADDRESS:IntegerConstant** **+ERR_INVALID_TELEPHONE_NUMBER:IntegerConstant** **+ERR_INVALID_UI_PARAMETERS:IntegerConstant** **+ERR_INVALID_INVALID_DATA:IntegerConstant** **+ERR_UI_PROBLEM:IntegerConstant** **+NRT_VCDMSSD:StructureDefinition** **+nrtVcdmSsd:NRT_VCDMSSD**
	+NRT_VCDM () **+~NRT_VCDM ()** **+nrt_vcdm ():Integer**
Purpose	Implements the NRT-VCDM SIB of the IIIN.
Description of attributes and operations	**O_SUCCESS:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_INVALID_INTERNET_ADDRESS:IntegerConstant,** **ERR_INVALID_TELEPHONE_NUMBER:IntegerConstant,** **ERR_INVALID_UI_PARAMETERS:IntegerConstant,** **ERR_INVALID_INVALID_DATA:IntegerConstant,** **ERR_UI_PROBLEM:IntegerConstant** : Error value constants. **NRT_VCDMSSD:StructureDefinition** : NRT-VCDM SIB Service Specific Data structure. **nrtVcdmSsd:NRT_VCDMSSD** : Holds the Service Specific Data.
	NRT_VCDM () : Constructor. **~NRT_VCDM ()** : Desctructor. **nrt_vcdm ():Integer** : Executes the NRT-VCDM SIB.

UML	**RCI**
	+O_SUCCESS:IntegerConstant **+O_ERROR:IntegerConstant** **+ERR_INVALID_ID:IntegerConstant** **+ERR_CALL_INFO_NOT_FOUND:IntegerConstant** **+RCISSD:StructureDefinition** **+rciSsd:RCISSD**
	+RCI () **+~RCI ()** **+rci ():Integer**
Purpose	Implements the RCI SIB of the IIIN.
Description of attributes and operations	**O_SUCCESS:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_INVALID_ID:IntegerConstant,** **ERR_CALL_INFO_NOT_FOUND:IntegerConstant** : Error value constants. **RCISSD:StructureDefinition** : RCI SIB Service Specific Data structure. **rciSsd:RCISSD** : Holds the Service Specific Data.
	RCI () : Constructor. **~RCI ()** : Desctructor. **rci ():Integer** : Executes the RCI SIB.

UML	**SSCR**
	+O_SUCCESS:IntegerConstant **+O_ERROR:IntegerConstant** **+ERR_INVALID_SC_TYPE:IntegerConstant** **+ERR_INVALID_ADDRESS:IntegerConstant** **+ERR_INVALID_PARAMETERS:IntegerConstant** **+SSCRSSD:StructureDefinition** **+sscrSsd:SSCRSSD**
	#SSCR () **+~SSCR ()** **+sscr ():Integer**
Purpose	Implements the SSCR SIB of the IIIN.
Description of attributes and operations	**O_SUCCESS:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_INVALID_SC_TYPE:IntegerConstant,** **ERR_INVALID_ADDRESS:IntegerConstant,** **ERR_INVALID_PARAMETERS:IntegerConstant** : Error value constants. **SSCRSSD:StructureDefinition** : SSCR SIB Service Specific Data structure. **sscrSsd:SSCRSSD** : Holds the Service Specific Data.
	SSCR () : Constructor. **~SSCR ()** : Desctructor. **sscr ():Integer** : Executes the SSCR SIB.

UML	**TIMER**
	+O_END:IntegerConstant **+O_ERROR:IntegerConstant** **+ERR_INVALID_TIME_VALUE:IntegerConstant** **+TPT_ABSOLUTE:IntegerConstant** **+TPT_RELATIVE:IntegerConstant** **+TIMERSSD:StructureDefinition** **+timerSsd:TIMERSSD**
	+TIMER () **+~TIMER ()** **+timer ():Integer**
Purpose	Implements the TIMER SIB of the IIIN.
Description of attributes and operations	**O_END:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_INVALID_TIME_VALUE:IntegerConstant** : Error value constants. **TPT_ABSOLUTE:IntegerConstant, TPT_RELATIVE:IntegerConstant** : Time value type constants. **TIMERSSD:StructureDefinition** : TIMER SIB Service Specific Data structure. **timerSsd:TIMERSSD** : Holds the Service Specific Data.
	TIMER () : Constructor. **~TIMER ()** : Desctructor. **timer ():Integer** : Executes the TIMER SIB.

UML	**CALL_LIST**
	+GOTNUM:IntegerConstant **+ALLDONE:IntegerConstant** **+TRYLATER:IntegerConstant** **+O_ERROR:IntegerConstant** **+ERR_IN_LIST_NOT_PRESENT:IntegerConstant** **+ERR_OUT_LIST_NOT_PRESENT:IntegerConstant** **+ERR_IN_LIST_INVALID:IntegerConstant** **+CALLLISTSSD:StructureDefinition** **+callListSsd:CALLLISTSSD**
	+CALL_LIST () **+~CALL_LIST ()** **+call list ():Integer**
Purpose	Implements the CALL LIST SDB of the IIIN.
Description of attributes and operations	**GOTNUM:IntegerConstant, ALLDONE:IntegerConstant, TRYLATER:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **ERR_IN_LIST_NOT_PRESENT:IntegerConstant, ERR_OUT_LIST_NOT_PRESENT:IntegerConstant, ERR_IN_LIST_INVALID:IntegerConstant** : Error value constants. **CALLLISTSSD:StructureDefinition** : CALL LIST SDB Service Specific Data structure. **callListSsd:CALLLISTSSD** : Holds the Service Specific Data.
	CALL_LIST () : Constructor. **~CALL_LIST ()** : Desctructor. **call list ():Integer** : Executes the CALL LIST SDB.

UML	USER_INTERACTION
	+O_SUCCESS:IntegerConstant +O_ERROR:IntegerConstant +PLAY_ANNOUNCEMENT:IntegerConstant +COLLECT_DIGITS:IntegerConstant +UISSD:StructureDefinition +uiSsd:UISSD
	+USER_INTERACTION () +~USER_INTERACTION () +user_interaction ():Integer
Purpose	Implements the USER INTERACTION SDB of the IIIN.
Description of attributes and operations	**O_SUCCESS:IntegerConstant, O_ERROR:IntegerConstant** : SIB outlet constants. **PLAY_ANNOUNCEMENT:IntegerConstant, COLLECT_DIGITS:IntegerConstant** : Action type constants. **UISSD:StructureDefinition** : USER INTERACTION SIB Service Specific Data structure. **uiSsd:UISSD** : Holds the Service Specific Data.
	USER_INTERACTION () : Constructor. **~USER_INTERACTION ()** : Desctructor. **user_interaction ():Integer** : Executes the USER INTERACTION SDB.

UML	END
	+END () +~END () +end ()
Purpose	Implements the END SDB of the IIIN.
Description of attributes and operations	**END ()** : Constructor. **~END ()** : Desctructor. **end ()** : Executes the END SDB.

UML	**SDF**
	-hSCFaccess:Handle **-hBufSemaphore:Handle** **-messageBuffer:OctetArray** **-fileList:FileArray**
	+SDF(); **+~SDF();** **+data (theData:OctetPointer)**
Purpose	Implements the SDF Functional Entity of the IN.
Description of attributes and operations	**hSCFaccess:Handle** : Event handle for signalling the SCF when the result is ready. **hBufSemaphore:Handle** : Semaphore on which the SDF is signalled when its out-buffer is available for use again after sending. **messageBuffer:OctetArray** : Buffer used for sending messages to SCF, with the aid of the above event and semaphore. **fileList:FileArray** : Array of pointers to files containing different database information.
	SDF() : Constructor. **~SDF()** : Destructor. **data (theData:OctetPointer)** : Processes a raw message from the SCF, contained in theData.

Objects of the iscgf Module Supporting Both Socket and CORBA Communication

UML	**ISCGF**
	-accessMgr:AccessManager **-tbp:IIIN_TBP** **-out data:ISCGFSCFData**
	+ISCGF() **+~ISCGF()** **+ExecuteServiceComponent** **(component_type:IIINServiceComponentType, parameters:Any)** **+run ()**
Purpose	Implements an ISCGF Functional Entity of the IIIN that supports both socket and CORBA communication.
Description of attributes and operations	**accessMgr:AccessManager** : An instance of a socket reader/writer. **tbp:IIIN_TBP** : An instance of a helper class which processes IIINTPB message data. **out_data:ISCGFSCFData** : A structure to hold the message for the SCF, before it is processed for sending.
	ISCGF () : Constructor. **~ISCGF ()** : Destructor. **ExecuteServiceComponent** **(component_type:IIINServiceComponentType, parameters:Any)** : The CORBA method for invoking services from the Internet. **run ()** : Main loop that waits for requests, both from the Internet and from the SCF.

Objects of the iμs Module Using Socket Communication

UML	**ImicroS_S** **-tbp:IIIN_TBP** **-listenSock:Socket** **-dataSock:Socket** **-browserPath:CharacterArray** **-fp:FilePointer** **+IMicroS_S (browser_path:CharacterPointer)** **+~IMicroS_S ()** **+run ()**
Purpose	Implements an IμS Functional Entity of the IIIN that supports socket communication.
Description of attributes and operations	**tbp:IIIN_TBP** : An instance of a helper class which processes IIINTPB message data. **listenSock:Socket** : Socket that listens for requests. **dataSock:Socket** : Socket for accepting requests and data. **browserPath:CharacterArray** : Path to the browser to be used. **fp:FilePointer** : Pointer to log file.
	IMicroS_S () : Constructor. **~ IMicroS_S ()** : Destructor. **run ()** : Main loop that waits for requests from the ISCGF.

Appendix G

Examples of C++, C and Java Code

A) C++ Code Implementing the Two SLPs of the IIIN User Registration Service Feature

The IIINRegistrationMain SLP is initiated from the Internet and is the main SLP of this service. It actually performs the registration i.e. the storing of the user's data in the database. IIINRegistrationConfirm is an assisting SLP. It is initiated from the SSF and performs the collection of the confirmation code. It passes the dialled code to the IIINRegistrationMain SLP through an instance of the CIU SIB. The IIINRegistrationMain SLP then checks if the code is correct and proceeds accordingly.

The functions make standard function calls and calls on SIB objects, which have been described in UML in Appendix F.

```
////////////////////////////////////////////////////////////////////
//*****************************************************************//
//*****************************************************************//
//********************** IIINRegistrationMain *********************//
//*****************************************************************//
//*****************************************************************//
////////////////////////////////////////////////////////////////////
/////
RTNCODE IIINRegistrationMain (void *pData)
{
   try {
      std::list<jstring> dataList;
      FRL::extract_iiin_slp_data (dataList,
                                  FRL::get_in_data (),
                                  FRL::get_in_data_len ());
      if (dataList.size () != 6) {
         EnterCriticalSection(evo);
         cerr << "IIINRegistrationMain: invalid data (1). Exiting."
              << endl;
         LeaveCriticalSection(evo);
         return FAILURE;
      }

//*****************************************************************//
// RCI                                                             //
//*****************************************************************//
      FRL::IIIN::RCI retrieveCallInfoObj;
      retrieveCallInfoObj.rciSsd.timeout = 600;
      jstring requiredCode (dataList.back ());
```

```
      // the required code is the last among the received parameters
      retrieveCallInfoObj.rciSsd.pcidValueToMatch = &requiredCode;
      jstring number (dataList.front ());
      retrieveCallInfoObj.rciSsd.pcidIdentifier = &number;
      jstring code;
      retrieveCallInfoObj.rciSsd.pcidDynamicInfo = &code;
      int errorInt;
      retrieveCallInfoObj.rciSsd.pcidError = &errorInt;
      int sibRet;
      if ((sibRet = retrieveCallInfoObj.rci ()) ==
          FRL::IIIN::RCI::O_ERROR) {
         EnterCriticalSection(evo);
         cerr << "IIINRegistrationMain: error in RCI SIB: errno = "
              << errorInt << endl;
         LeaveCriticalSection(evo);
         return FAILURE;
      }
// End RCI                                                          //
//******************************************************************//

      jstring sscrParameters;
      jstring ipAddress;
      // check if the call information was found and if it is valid
      if ((sibRet == O_FAILURE) ||
          (sibRet == O_TIMEOUT)){

//******************************************************************//
// SDM (loop)                                                       //
//******************************************************************//
         char filename[30] = "IIINUser_NUM_";
         if (dataList.front ().size () > 15) {
            EnterCriticalSection(evo);
            cerr << "IIINRegistrationMain: invalid data (1). "
                 << "Exiting." << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
         strcat (filename, dataList.front ().c_str ());
         dataList.pop_front ();

         SDMSSD sdmSsd;
         sdmSsd.file = filename;
         sdmSsd.action = REPLACE;
         char info[61];
         sdmSsd.pcidInfo = info;
         int errorInt;
         sdmSsd.pcidError = &errorInt;

         unsigned long eis[3] = {SDMSSD::NAME,
                                 SDMSSD::IP,
                                 SDMSSD::EMAIL};
         for (int i = 0; i < 3; i++) {
            if (i == 1) {
               if (dataList.front () != "0") {
                  dataList.pop_front ();
                  continue;
               }
               else {
                  dataList.pop_front ();
                  ipAddress = dataList.front ();
               }
            }
            sdmSsd.elementIndicator = eis[i];
            if (dataList.front ().size () > 60) {
```

```
            EnterCriticalSection(evo);
            cerr << "IIINRegistrationMain: invalid data (" << i
                 << "). Exiting." << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
         strcpy (info, dataList.front ().c_str ());

         if(FRL::call_SDM (&sdmSsd) == ERROR) {
            EnterCriticalSection(evo);
            cerr << "IIINRegistrationMain: error in SDM SIB ("
                 << i << "), errno = " << errorInt
                 << ". Exiting." << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
         dataList.pop_front ();
      }
// End SDM (loop)                                                    //
//*****************************************************************//

      // set confirmation html page as parameter for SSCR
      // (SK_DISPLAY)
      sscrParameters =
        "http://www.teltec.dcu.ie/~vasicj/mag/user_regconf.html";
   }
   else {
      // pop the name
      dataList.pop_front ();
      dataList.pop_front ();
      if (dataList.front () == "0") {
         dataList.pop_front ();
         ipAddress = dataList.front ();
      }
      // set confirmation html page as parameter for SSCR
      //(SK_DISPLAY)
      sscrParameters =
     "http://www.teltec.dcu.ie/~vasicj/mag/user_reginvcode.html";
   }

//*****************************************************************//
// SSCR                                                             //
//*****************************************************************//
   FRL::IIIN::SSCR sscrObj;
   sscrObj.sscrSsd.serviceComponentType = SK_DISPLAY;
   if (ipAddress == "") {
      // do whatever is needed to get the dynamic ip address...
   }
   sscrObj.sscrSsd.pcidAddress = &ipAddress;
   sscrObj.sscrSsd.pcidServiceComponentParameters =
                                              &sscrParameters;
   sscrObj.sscrSsd.pcidError = &errorInt;

   if (sscrObj.sscr () == FRL::IIIN::SSCR::O_ERROR) {
      EnterCriticalSection(evo);
      cerr << "IIINRegistrationMain: error in SSCR SIB, errno = "
           << errorInt;
      cerr << ". Exiting." << endl;
      LeaveCriticalSection(evo);
      return FAILURE;
   }
// End SSCR                                                          //
//*****************************************************************//
```

```
//*****************************************************************//
// END                                                             //
//*****************************************************************//
      FRL::CS_2::END e;
      e.end ();
// End END                                                         //
//*****************************************************************//
   }
   catch (FRL::SystemException & e) {
      cerr << "IIINRegistrationMain: " << e.text << endl;
      return FAILURE;
   }
   catch (...) {
      cerr << "IIINRegistrationMain: unknown exception!" << endl;
      return FAILURE;
   }

   return SUCCESS;
}


////////////////////////////////////////////////////////////////////
//*****************************************************************//
//*****************************************************************//
//****************** IIINRegistrationConfirm SLP ******************//
//*****************************************************************//
//*****************************************************************//
////////////////////////////////////////////////////////////////////
RTNCODE IIINRegistrationConfirm (void *pData)
{
   try {
//*****************************************************************//
// UI (1)                                                          //
//*****************************************************************//
      FRL::CS_1::USER_INTERACTION uiObj;
      uiObj.uiSsd.uiType =
         FRL::CS_1::USER_INTERACTION::COLLECT_DIGITS;
      jstring digits;
      uiObj.uiSsd.pcidDigits = &digits;
      int errorInt;
      uiObj.uiSsd.pcidError = &errorInt;
      if (uiObj.user_interaction () ==
            FRL::CS_1::USER_INTERACTION::O_ERROR) {
         EnterCriticalSection(evo);
         cerr << "IIINRegistrationConfirm: error in UI SIB (1), "
              << "errno = " << errorInt
              << ". Exiting." << endl;
         LeaveCriticalSection(evo);
         return FAILURE;
      }
// End UI (1)                                                      //
//*****************************************************************//

//*****************************************************************//
// UI (2)                                                          //
//*****************************************************************//
      uiObj.uiSsd.uiType =
         FRL::CS_1::USER_INTERACTION::PLAY_ANNOUNCEMENT;
      jstring announcement ("The digits you have entered have been"
                            "collected. Thank you.");
      uiObj.uiSsd.pcidAnnouncement = &announcement;
      uiObj.uiSsd.pcidDigits = NULL;
      if (uiObj.user_interaction () ==
            FRL::CS_1::USER_INTERACTION::O_ERROR) {
```

```
            EnterCriticalSection(evo);
            cerr << "IIINRegistrationConfirm: error in UI SIB (2), "
                 << "errno = " << errorInt
                 << ". Exiting." << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
// END UI (2)                                                          //
//*********************************************************************//

//---------------------------------------------------------------------//
// extract incoming data                                               //
//---------------------------------------------------------------------//
         UBYTE * data_p = FRL::get_in_data ();
         if ((data_p[3] != ARGTAG_CALLING_LINE_ID) ||
             (data_p[4] > 15)) {
            EnterCriticalSection(evo);
            cerr << "GroupAnnouncementsRec: invalid data (1)!" << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
         jstring callingNumber;
         for (int i = 0; i < data_p[4]; i++) {
            callingNumber.append ((char)(data_p[5 + i] + 48));
         }

//*********************************************************************//
// CIU                                                                 //
//*********************************************************************//
         FRL::IIIN::CIU ciu;
         ciu.ciuSsd.pcidIdentifier = &callingNumber;
         ciu.ciuSsd.pcidDynamicInfo = &digits;
         ciu.ciuSsd.timeout = 600;
         ciu.ciuSsd.pcidError = &errorInt;

         if (ciu.add_ci () == FRL::IIIN::CIU::O_ERROR) {
            EnterCriticalSection(evo);
            cerr << "IIINRegistrationConfirm: error in CIU SIB, "
                 << "errno = " << errorInt << ". Exiting." << endl;
            LeaveCriticalSection(evo);
            return FAILURE;
         }
// End CIU                                                             //
//*********************************************************************//
      }
      catch (FRL::SystemException & e) {
         cerr << "IIINRegistrationConfirm: " << e.text << endl;
         return FAILURE;
      }
      catch (...) {
         cerr << "IIINRegistrationConfirm: unknown exception!" << endl;
         return FAILURE;
      }

      return SUCCESS;
}
```

B) C Code Listing for the iiin_user_reg CGI Program, which Processes a User's Request for IIIN Registration

The following code is built into a CGI program that processes the HTML IIIN User Registration Form. The main function makes calls to the following functions:

- print_system_error_message () – creates and displays html page advising the user that a processing error occurred
- print_invalid_data_message () – creates and displays html page warning the user that some of the data entered in the form is invalid
- verify_ip_address () – verifies the format of the entered IP address
- verify_email_address () – verifies the format of the entered email address
- verify_number () – verifies the format of the entered telephone number
- remove_percentage_chars () – removes the '%' characters from the string received from the WWW server (these are used in that string to code some special characters)
- send_iiin_tbp_message () – sends a prepared IIINTBP message to the ISCGF

```
main (int argc, char * argv[])
{
  char * buffer;
  char * endOfData;
  char * contentLength;
  int length;
  char * p;
  int fieldCounter = 0;
  int currentIsName = 1;
  int i, j;
  char codeString[CODE_STRING_LENGTH + 1];
  int ret;
  char buf[512];
  unsigned char sendIP = 1;
  int ret_length;
  char * indfile_path = "/storage4/IN/vasicj/indfiles/";
  char indfile_name[50];
  FILE * test_fd;
  struct dirent * dirent_p;
  DIR * indfiledirp;
  struct stat fileent;


  /* first get rid of old indicator files (these indicator files are
     empty files, created for each user at registration and kept for
     15 minutes; the presence of such a file for a user prevents
     him/her to register - see in program below)*/
  indfiledirp = opendir ("/storage4/IN/vasicj/indfiles");
  if (!indfiledirp) {
    print_system_error_message (1);
    return 1;
  }
  while (dirent_p = readdir (indfiledirp)) {
    if (strcmp (dirent_p->d_name, ".") &&
        strcmp (dirent_p->d_name, "..")) {
      sprintf (indfile_name, "%s%s", indfile_path,
```

```
            dirent_p->d_name);
    if (stat (indfile_name, &fileent) == -1) {
    temp_buf1 = indfile_name;
    print_system_error_message (2);
    return 1;
    }
    if (time (NULL) - fileent.st_mtime > 900) {
    if (remove (indfile_name) == -1) {
      print_system_error_message (3);
      return 1;
    }
    }
  }
}
if (errno == EBADF) {
  print_system_error_message (4);
  return 1;
}
if (closedir (indfiledirp) == -1) {
  print_system_error_message (5);
  return 1;
}

/* get length of data */
contentLength = (char *)getenv ("CONTENT_LENGTH");
if (!contentLength) {
  print_system_error_message (6);
  return 1;
}

if (!sscanf (contentLength, "%d", &length)) {
  print_system_error_message (7);
  return 1;
}

/* prepare buffer for data */
buffer = (char *)malloc (length + 1);
if (buffer == NULL) {
  print_system_error_message (8);
  return 1;
}

/* read data into buffer */
if (fread (buffer, 1, length, stdin) != length) {
  print_system_error_message (9);
  return 1;
}

ret_length = remove_percentage_chars (buffer, length);
if (ret_length < 0) {
  print_system_error_message (2000 - ret_length);
  return 1;
}
length = ret_length;

endOfData = buffer + length;
p = buffer;

/* parse data received from server*/
while (1) {
  if (fieldCounter > MAX_FIELDS) {
    print_system_error_message (10);
    return 1;
  }
```

```
  if (currentIsName) {
    fieldNames_p[fieldCounter] = p;
  }
  else {
    fieldValues_p[fieldCounter++] = p;
  }

  while (p < endOfData) {
    if (*p == (currentIsName ? '=' : '&')) {
    *p = '\0';
    p++;
    break;
    }
    p++;
  }

  if (p == endOfData) {
    if (currentIsName) {
    if (*(p-1)) {
      print_system_error_message (11);
      return 1;
    }
    fieldValues_p[fieldCounter++] = p;
    currentIsName = 0;
    }
    *p = '\0';
    break;
  }
  else {
    currentIsName = 1 - currentIsName;
  }
}

/* examine data */
for (i = 0; i < fieldCounter; i++) {
  if (fieldNames_p[i] && fieldValues_p[i]) {
    if (!strcmp (fieldNames_p[i], "name")) {
      if (strcmp (fieldValues_p[i], "")) {
        isPresent[NAME_INDEX] = 1;
        indices[NAME_INDEX] = i;
      }
    }
    else if (!strcmp (fieldNames_p[i], "number")) {
      if (verify_number (i)) {
        isPresent[NUMBER_INDEX] = 1;
        indices[NUMBER_INDEX] = i;
      }
    }
    else if (!strcmp (fieldNames_p[i], "ipaddralloctype")) {
      /* must be OK */
      isPresent[ALLOCTYPE_INDEX] = 1;
      indices[ALLOCTYPE_INDEX] = i;
    }
    else if (!strcmp (fieldNames_p[i], "ipaddress")) {
      if (verify_ip_address (i)) {
        isPresent[IPADDRESS_INDEX] = 1;
        indices[IPADDRESS_INDEX] = i;
      }
    }
    else if (!strcmp (fieldNames_p[i], "email")) {
      if (verify_email_address (i)) {
        isPresent[EMAIL_INDEX] = 1;
        indices[EMAIL_INDEX] = i;
```

```
        }
      }
    }
    else {
      print_system_error_message (12);
      return 1;
    }
  }

  if (!strcmp (fieldValues_p[ALLOCTYPE_INDEX], "dynamic")) {
    isPresent[IPADDRESS_INDEX] = 1;
    sendIP = 0;
  }

  /* find if any of the data is missing or invalid */
  fieldCounter = 0;
  for (i = 0; i < MAX_FIELDS; i++) {
    if (!isPresent[i]) fieldCounter++;
  }
  if (fieldCounter) {
    print_invalid_data_message (fieldCounter);
    return 1;
  }

  /* form 5-digit random number */
  srand (gethrtime () % UINT_MAX);
  for (i = 0; i < CODE_STRING_LENGTH; i++) {
    codeString[i] = (int)(((float)rand ()/(float)32768) * 10)
                                                        + 0x30;
  }
  codeString[CODE_STRING_LENGTH] = '\0';

  /* form the indicator file name */
  sprintf (indfile_name, "%s%s%s", indfile_path, "if",
           fieldValues_p[indices[NUMBER_INDEX]]);

  /* if the indicator file is present, registration is not
     possible */
  test_fd = fopen (indfile_name, "r");
  if (test_fd) {
    printf ("Content-type: text/html\n\n");
    printf ("<html><head>\n");
    printf ("<title>IIIN User Registration Error - Already"
            "Registered</title>");
    printf ("</head>\n");
    printf ("<body>\n");
    printf ("<h2><font color=\"#FF0000\">Sorry, you have already
registered once in the last 15 minutes. If you wish to register once
more, in order to change your data, please wait for 15 minutes and
then try again.<br></h2>\n");
    printf ("</body></html>\n");
    fflush (stdout);
    return 0;
  }
  /* all is well, create indicator file */
  test_fd = fopen (indfile_name, "w");
  if (!test_fd) {
    print_system_error_message (13);
    return 1;
  }
  fprintf (test_fd, "\n");
  if (fclose (test_fd) == EOF) {
    print_system_error_message (14);
    return 1;
```

```
}

/* form the IIIN TBP message */
sprintf (buf,
        "IIINREQUEST\n"
        "COMPONENT %d\n"
        "PARAMETERS %d\n"
        "CG_NUM %s\n"
        "NAME %s\n"
        "DYN_IP %d\n"
        "%s%s%s"
        "EMAIL %s\n"
        "CODE %s\n"
        "PARAMETERS END\n"
        "IIINREQUEST END\n",
        SK_CPM_IIIN_REGISTRATION,                  /* component */
        isPresent[IPADDRESS_INDEX] ? 6 : 5,        /* parameters */
        fieldValues_p[indices[NUMBER_INDEX]],      /* cg_num */
        fieldValues_p[indices[NAME_INDEX]],        /* name */
        1 - sendIP,                                /* DYN_IP */
        /* the next 3 lines compose the IP address, if present */
        sendIP ? "IP " : "",
        sendIP ? fieldValues_p[indices[IPADDRESS_INDEX]] : "",
        sendIP ? "\n" : "",
        fieldValues_p[indices[EMAIL_INDEX]],       /* name */
        codeString);

/* send message to the ISCGF */
ret = send_iiin_tbp_message (buf, strlen (buf));
if (ret) {
  print_system_error_message (1000 + ret);
  return 0;
}

/* instruct the user to enter the code through the telephone */
printf ("Content-type: text/html\n\n");
printf ("<html><head>\n");
printf ("<title>IIIN User Registration Form "
        "Instructions</title>");
printf ("</head>\n");
printf ("<body>\n");
printf ("<h2><font color=\"#FF0000\">Thank you for your interest "
        "in IIIN.<br></h2>\n");
printf ("<h3>In order to complete the registration process, "
      "please dial #00 from your phone and at the prompt enter "
      "the digits %s<br></h3>\n", codeString);
printf ("</body></html>\n");
fflush (stdout);

free (buffer);

return 0;
}
```

C) Java Code Listing for the CTDB (Click-to-Dial-Back) Applet

This applet is part of the HTML page shown in Figure 5.11. It is used to request the Click-to-Dial-Back service. The applet sends a request for the service directly to the ISCGF in a CORBA method invocation. The classes of the IIN module used here have been generated by the Visibroker IDL to Java compiler from the IDL shown in Appendix D.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import org.omg.CORBA.*;

public class CTDB extends Applet implements MouseListener {
  static final String firstLine = "Click here and we";
  static final String secondLine = "will call you back!";
  static final String waitLine = "Please wait...";

  private Font font;
  private String ipAddress;
  private ORB orb = null;
  private IIIN.InternetSCFGateway iscgf = null;
  private boolean clicked;

  public void init () {
    font = new Font ("Helvetica", Font.BOLD, 14);

    this.addMouseListener (this);

    try {
      orb = ORB.init (this, null);
      iscgf = IIIN.InternetSCFGatewayHelper.bind (orb, "IIINISCGF");
      ipAddress = InetAddress.getLocalHost ().getHostAddress ();
      clicked = false;
    }
    catch (SystemException se) {
      Graphics g = getGraphics ();
      g.setColor (Color.blue);
      g.fillOval (1, 1, 166, 60);
    }
    catch (UnknownHostException uhe) {
      Graphics g = getGraphics ();
      g.setColor (Color.yellow);
      g.fillOval (1, 1, 166, 60);
    }
  }

  public void paint (Graphics g) {
    g.setColor (Color.white);
    g.fillRect(0, 0, 168, 62);

    g.setColor (Color.black);
    g.fillOval (1, 1, 166, 60);

    g.setColor (Color.lightGray);
    g.fillOval (5, 5, 158, 52);

    g.setColor (Color.black);
    g.setFont (font);
```

```
    g.drawString (firstLine, 22, 28);
    g.drawString (secondLine, 23, 45);
  }

  public void mouseEntered (MouseEvent me) {;}
  public void mouseExited (MouseEvent me) {;}
  public void mousePressed (MouseEvent me) {;}
  public void mouseReleased (MouseEvent me) {;}

  public void mouseClicked (MouseEvent me) {

    if (clicked == false) {
      clicked = true;

      // change the graphics
      Graphics g = getGraphics ();

      g.setColor (Color.lightGray);
      g.fillOval (5, 5, 158, 52);

      g.setColor (Color.black);
      g.setFont (font);
      g.drawString (waitLine, 41, 35);

      // then make the CORBA call...
      if (me.getID () == me.MOUSE_CLICKED) {
         // the number of the service subscriber is hardcoded into
         // the method (not the best way of doing it but the
         //fastest)
         String numberToCall = getParameter ("NumberToCall");
        IIIN.ClickToDialBackData ctdbd =
          new IIIN.ClickToDialBackData (ipAddress, numberToCall);
        Any params = orb.create_any ();
        IIIN.ClickToDialBackDataHelper.insert(params, ctdbd);
        iscgf.ExecuteServiceComponent
           (IIIN.CLICK_TO_DIAL_BACK.value, params);
      }
    }
  }
}
```