A Structural and Functional Specification of a SCIM for Service Interaction Management and Personalisation in the IMS

Submitted in fulfilment
of the requirements of the degree
Doctor of Philosophy
at Rhodes University

Mosiuoa Tsietsi

August 2011

Abstract

The Internet Protocol Multimedia Subsystem (IMS) is a component of the 3G mobile network that has been specified by standards development organisations such as the 3GPP (3rd Generation Partnership Project) and ETSI (European Telecommunication Standards Institute). IMS seeks to guarantee that the telecommunication network of the future provides subscribers with seamless access to services across disparate networks. In order to achieve this, it defines a service architecture that hosts application servers that provide subscribers with value added services.

Typically, an application server bundles all the functionality it needs to execute the services it delivers, however this view is currently being challenged. It is now thought that services should be synthesised from simple building blocks called service capabilities. This decomposition would facilitate the re-use of service capabilities across multiple services and would support the creation of new services that could not have originally been conceived.

The shift from monolithic services to those built from service capabilities poses a challenge to the current service model in IMS. To accommodate this, the 3GPP has defined an entity known as a service capability interaction manager (SCIM) that would be responsible for managing the interactions between service capabilities in order to realise complex services. Some of these interactions could potentially lead to undesirable results, which the SCIM must work to avoid. As an added requirement, it is believed that the network should allow policies to be applied to network services which the SCIM should be responsible for enforcing. At the time of writing, the functional and structural architecture of the SCIM has not yet been standardised.

This thesis explores the current service architecture of the IMS in detail. Proposals that address the structure and functions of the SCIM are carefully compared and contrasted. This investigation leads to the presentation of key aspects of the SCIM, and provides solutions that explain how it should interact with service capabilities, manage undesirable interactions and factor user and network operator policies into its execution model. A modified design of the IMS service layer that embeds the SCIM is subsequently presented and described. The design uses existing IMS protocols and requires no change in the behaviour of the standard IMS entities.

In order to develop a testbed for experimental verification of the design, the identification of suitable software platforms was required. This thesis presents some of the most popular platforms currently used by developers such as the Open IMS Core and OpenSER, as well as an

open source, Java-based, multimedia communication platform called Mobicents. As a precursor to the development of the SCIM, a converged multimedia service is presented that describes how a video streaming application that is leveraged by a web portal was implemented for an IMS testbed using Mobicents components. The Mobicents SIP Servlets container was subsequently used to model an initial prototype of the SCIM, using a multi-component telephony service to illustrate the proposed service execution model. The design focuses on SIP-based services only, but should also work for other types of IMS application servers as well.

Acknowledgements

I wish to thank my supervisors, Prof. Alfredo Terzoli and Prof. George Wells for their tireless effort and patience towards me over the past three years. This thesis bears testimony to the quality of their stewardship and their ability to motivate students to do their best.

I would also like to thank the staff members and students of the Department of Computer Science at Rhodes University, especially the past and present members of the Convergence Research Group.

This thesis is dedicated to my family who have stood by me through thick and thin and believed in me through personal and professional challenges. I am grateful for the support they gave me not just over the past three years but all throughout my life. Particularly, I would like to thank my younger brother, who in his short life, taught me all I needed to know about the value of persistence and belief in oneself.

I am indebted to my circle of friends who constantly encouraged me in my efforts, while reminding me that there is more to life than just research.

A very special thank you to Teboho Mosuoe for the care and support she gave throughout this long and arduous process.

Finally, thank you to my sponsors: Telkom SA, Comverse, Stortech, Tellabs, Easttel, Bright Ideas 39 and THRIP who generously provided the financial support that allowed me to complete this work.

Contents

List of Figures

List of Tables

Glossary Of Terms

2G: Second Generation.

3G: Third Generation.

3GPP: 3rd Generation Partnership Project.

3GPP2: 3rd Generation Partnership Project 2.

3PCC: 3rd Party Call Controller.

AAA: Authentication, Authorisation and Accounting.

AGI: Asterisk Gateway Interface.

ARIB: Association of Radio Industries and Business.

AS: Application Server.

ATIS: Alliance for Telecommunication Industry Solutions.

ATM: Asynchronous Transfer Mode.

AUID: Application Unique Identifier.

B2BUA: Back-to-Back User Agent.

CAMEL: Customized Applications for Mobile network Enhanced Logic.

CAP: CAMEL Application Part.

CDMA2000: Code Division Multiple Access 2000.

CDR: Call Detail Record.

CN: Core Network.

CSCF: Call Session Control Function.

DECT: Digital Enhanced Cordless Communications.

DNS: Domain Name System.

EDGE: Enhanced Data Rates for GSM Evolution.

EISL: Extended IMS Service Layer.

ETSI: European Telecommunication Standards Institute.

FHoSS: FOKUS Home Subscriber Server.

FMJ: Freedom for Media in Java.

FPLMTS: Future Public Land Mobile Telecommunication System.

FQDN: Fully Qualified Domain Name.

FTP: File Transfer Protocol.

GPL: General Public License.

GSM: Global System for Mobile communication.

HSQLDB: Hyper Structured Query Language Database.

HTTP: Hyper Text Transfer Protocol.

I-CSCF: Interrogating CSCF.

IAB: Internet Architecture Board.

IANA: Internet Assigned Numbers Authority.

IC: Integrated Chip.

ICT: Information and Communication Technology.

IESG: Internet Engineering Steering Group.

IETF: Internet Engineering Task Force.

iFC: Initial Filter Criteria.

IM-SSF: IP Multimedia Service Switching Function.

IMPI: IP Multimedia Private Identity.

IMPU: IP Multimedia Public Identity.

IMS: IP Multimedia Subsystem.

IMT-2000: International Mobile Telecommunication in the year 2000.

IP: Internet Protocol.

IP CAN: IP Connectivity Access Network.

IPSec: IP Security.

IPTV: IP Television.

IPv4: IP version 4.

IPv6: IP version 6.

ISDN: Integrated Services Digital Network.

JAIN: Java APIs for Intelligent Networks.

JAIN SLEE: JAIN Service Logic and Execution Environment.

Java EE: Java Enterprise Edition.

Java SE: Java Standard Edition.

JCP: Java Community Process.

JMF: Java Media Framework.

JMX: Java Management Extensions.

JSR: Java Specification Request.

LAN: Local Area Network.

LIF: Location Interoperability Forum.

MGCP: Media Gateway Control Protocol.

MGIF: Mobile Gaming Interoperability Forum.

MMD: Multimedia Domain.

MMS: Multimedia Messaging Service.

MRF: Media Resource Function.

MRFC: Media Resource Function Controller.

MRFP: Media Resource Function Processor.

MT: Mobile Termination.

MWIF: Mobile Wireless Internet Forum.

NGN: Next Generation Network.

OMA: Open Mobile Alliance.

OSA: Open Service Access.

OSA SCS: OSA Service Capability Server.

OSS: Operations Support System.

P-CSCF: Proxy CSCF.

PBX: Private Branch Exchange.

PCG: Project Co-ordination Group.

PIDF: Presence Information Data Format.

PLMN: Public Land and Mobile Network.

PoC: Push to Talk over Cellular.

PSTN: Public Switched Telephone Network.

RA: Resource Adaptor.

RF: Radio Frequency.

RFC: Request for Comments.

S-CSCF: Serving CSCF.

SBB: Service Building Block.

SCIM: Service Capability Interaction Manager.

SDO: Standards Development Organisation.

SDP: Session Description Protocol.

SEMS: SIP Express Media Server.

SER: SIP Express Router.

SIPPING: SIP Protocol INvestiGation.

SLF: Subscription Locator Function.

SNR: Subscription Notification Request.

SPT: Service Point Trigger.

TE: Terminal Equipment.

TISPAN: Telecommunications and Internet converged Services and Protocols for Advanced Networking.

TR: Technical Report.

TS: Technical Specification.

TSG: Technical Specification Group.

TTA: Telecommunications Technology Association.

TTC: Telecommunication Technology Committee.

TTS: Text To Speech.

UDR: User Data Request.

UICC: Universal Integrated Circuit Card.

URI: Uniform Resource Identifier.

URL: Uniform Resource Locator.

UTRA: Universal Terrestrial Radio Access.

VoIP: Voice over IP.

VVoIP: Voice and Video over IP.

W3C: World Wide Web Consortium.

XCAP: XML Configuration AccessProtocol.

XDMS: XML Document Management Server.

XML: eXtensible Markup Language.

Chapter 1

Introduction

The Internet, a global overlay network for communications, has undergone numerous developments since its inception in the latter half of the 20th century. Arising from academic and military backgrounds, it has grown and spread in influence, becoming available to the majority of the population in developed countries and exhibiting increasing levels of penetration in the developing world as well [17]. One of the chief driving forces behind the success of the Internet is the range of diverse services that are available. At first, non-realtime services like email, FTP (File Transfer Protocol) and Usenet came into being. As new uses for the Internet as a service platform came into existence, realtime applications such as Voice and Video over the Internet Protocol (VVoIP) became rife, birthing popular online services like Skype and Google Talk. Currently, social networking, location based services and cloud computing have risen in popularity, and combined, have the potential to further revolutionise the way people use the Internet to interact with each other.

While the Internet steadily advanced, mobile and fixed line telecommunication networks were undergoing radical changes themselves. Though they remained tied to circuit-switching, fixed-line operators began providing ISDN (Integrated Services Digital Network) and subsequently DSL (Digital Subscriber Line) as new access technologies, the aim being to better support data and voice on the same network. Mobile operators, bolder than their contemporaries, went from the circuit-switched 2G (second generation) system to a packet-switched 3G (third generation) system, utilising packet-oriented protocols such as GPRS (General Packet Radio Service), EDGE (Enhanced Data rates for GSM Evolution) and HSDPA (High Speed Data Packet Access). The advent of these mobile broadband technologies had two effects: they supported the trend towards a packet-switched network that could support the Internet Protocol (IP); and they stimulated interest among the mobile subscriber base towards bandwidth-intensive, realtime communication services that could be supported much better due to faster connections.

The result has been increased availability and marketability of multimedia applications for communication and the diversification of services which now range from text-based chat applications to multi-party video conferencing applications. Previously, proponents of circuit-switching competed for control of the Internet through technologies such as ISDN and ATM (Asynchronous Transfer Mode). These proposals were met with resistance from proponents of packet-switching, who believed that IP was the most suitable candidate for supporting the Internet. History shows that this debate, now known as the battle between the Bell Heads (circuit-switching) versus the Net Heads (packet-switching) was won by the latter.

The flexibility, power and popularity of IP attracted the interest of standards development organisations (SDOs) who have sought to integrate IP into the 3G mobile system as an extension of the existing network by means of an all-IP core. This all-IP core would eventually come to be known as the IP Multimedia Subsystem, or IMS [18]. This native IP presence would help operators reduce capital expenditure and operating costs by providing a common backbone for all types of wireless access [19]. An all-IP core was also attractive as one of the main enablers for the realisation of service convergence in a Next Generation Network (NGN), providing connectivity to both mobile and fixed-line clients.

As a result of the need to incorporate IP into the evolved mobile network, SDOs involved in the development of technical specifications for 3G have had to liaise with their counterparts in the Internet domain. The IETF (Internet Engineering Task Force) and the OMA (Open Mobile Alliance) are not the only standards bodies involved in the evolution of Internet technology, but have had the strongest links with 3G standardisation, and continue to liaise with the SDOs on matters regarding the use of IP-based protocols in the IMS.

1.1 Setting the Stage for IMS

The International Telecommunication Union (ITU) is the leading United Nations agency for information and communication technology (ICT) issues and is the global focal point for governments and the private sector in developing networks and services [20]. The ITU has existed for over 140 years and during this period it has been instrumental in tackling issues such as the allocation of radio spectrum, the improvement of telecommunication infrastructure in developing countries across the world, the strengthening of cyber-security and the establishment of global standards that support many of the communication systems that are in use today. The ITU is divided into three main sectors, namely the ITU-R, the ITU-T and the ITU-D. The ITU-R is responsible for the management of the global RF (Radio Frequency) spectrum as well as the management of satellite orbit resources. The ITU-T is responsible for the development of global standards for ICT. The development of standards is what the ITU is primarily known for and is its oldest function. The ITU-D is responsible for issues pertaining to digital inclusion and promotes equitable, sustainable and affordable access to ICTs.

The ITU set the stage for the move to 3G mobile systems when it introduced its IMT-2000 (International Mobile Telecommunications in the year 2000) initiative. Formerly known as FPLMTS (Future Public Land Mobile Telecommunication Systems), IMT-2000 was birthed out of a desire to provide universal mobile coverage and seamless roaming abilities to mobile subscribers, with the added benefit of faster data rates to support a wider range of multimedia services [21]. In a recommendation published in 1997, the ITU solicited proposals for radio transmission technologies that would satisfy the requirements mandated by IMT-2000 [22]. Among the requirements was the need to provide data rates of 144kb/s for vehicular, 384kb/s for pedestrian and 2Mb/s for indoor environments. In response to this call, existing standardisation bodies in different parts of the world formed partnerships to work together to develop mobile standards for a 3G system that was in compliance with IMT-2000. Thus the IMS came into being, as a crucial middleware component of the 3G mobile system.

1.2 Standardisation in the IMS

IMS is a complex platform for many reasons. It must facilitate communication with non-IP networks such as the PSTN (Public Switched Telephone Network) and legacy mobile networks. It must reconcile differences between IPv4 (IP version 4) and IPv6 (IP version 6) domains and interconnect appropriately with different types of wireless local area networks. It is also responsible for providing seamless security, quality of service and billing capabilities across all communication sessions. Still, one of the main reasons why it is so complex is that it is in a constant state of flux.

From its humble beginnings in 1999 until now, the IMS has been evolving in order to meet new and emerging requirements. These requirements, and the solutions that were developed to address them, originated from governments and international organisations with vested interests in the IMS. Stakeholders would have the responsibility to plan the successful migration of their own established networks to an IMS platform. This was compounded by the fact that they each had to approach this migration from a different perspective, depending on whether they were fixed-line operators, mobile operators, service providers or national regulatory authorities. The inevitable result was the emergence of several different IMS flavours. This initially created interoperability problems and led to the duplication of efforts, which would have threatened to retard the adoption of IMS in the long run. In order to manage these and other challenges, healthy cooperation was cultivated between the various parties involved. This has allowed a more streamlined development of IMS standards, leading to the definition of a common set of features, which has come to be known as 'Common IMS'. In this chapter, a detailed inspection of the major SDOs in the area of IMS is undertaken and their participation in the move towards a Common IMS will be highlighted.

1.2.1 Main Standardisation Bodies

1.2.1.1 ETSI

The European Telecommunication Standards Institute (ETSI), much like the ITU, is a major development organisation that produces technical specifications covering a wide range of technologies such as fixed-line, mobile, radio, broadcast and the Internet [23]. ETSI was formed in 1988 to primarily focus on European requirements and European markets. Another strategic area of interest for them is in service provision, where ETSI provides services to ETSI members and other organisations in the form of hosting interoperability and protocol tests. ETSI is composed of a General Assembly which makes high-level decisions, a board which is the executive arm of the General Assembly, technical committees, ETSI projects and ETSI Partnership projects, and lastly a Secretariat. The bulk of the technical work is carried out in technical committees such as TC TISPAN (Technical Committee on Telecoms and Internet Converged Services and Protocols for Advanced Networks) and TC DECT (Technical Committee on Digital Enhanced Cordless Communications). Figure 1.1 shows the structure of ETSI.

1.2.1.2 3GPP

The 3rd Generation Partnership Project (3GPP) was created in 1998 by ETSI. 3GPP was formed to develop technical specifications and technical reports for a 3G mobile system based on an evolved GSM (Global System for Mobile communication) network and radio access networks

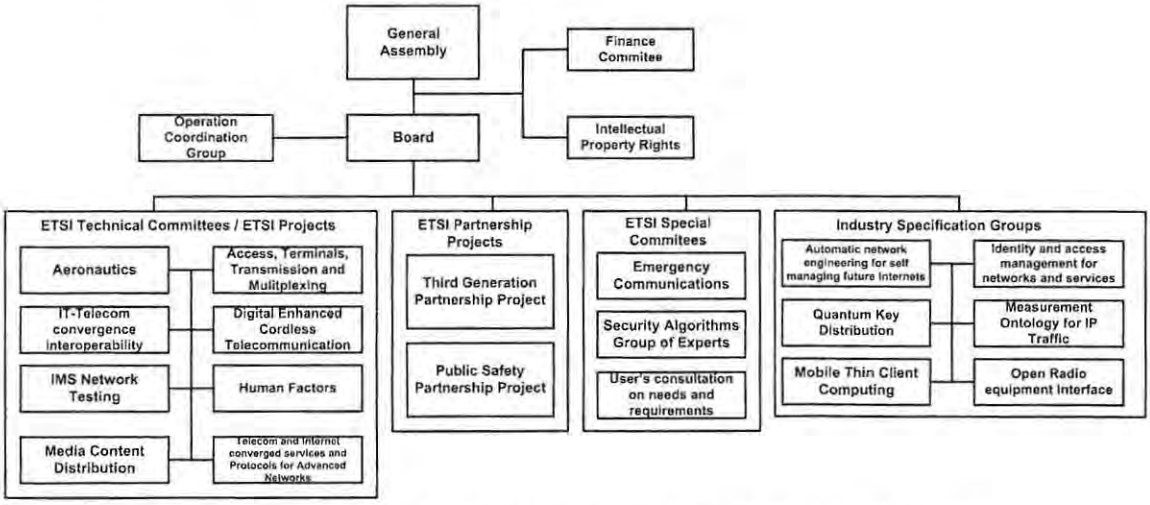
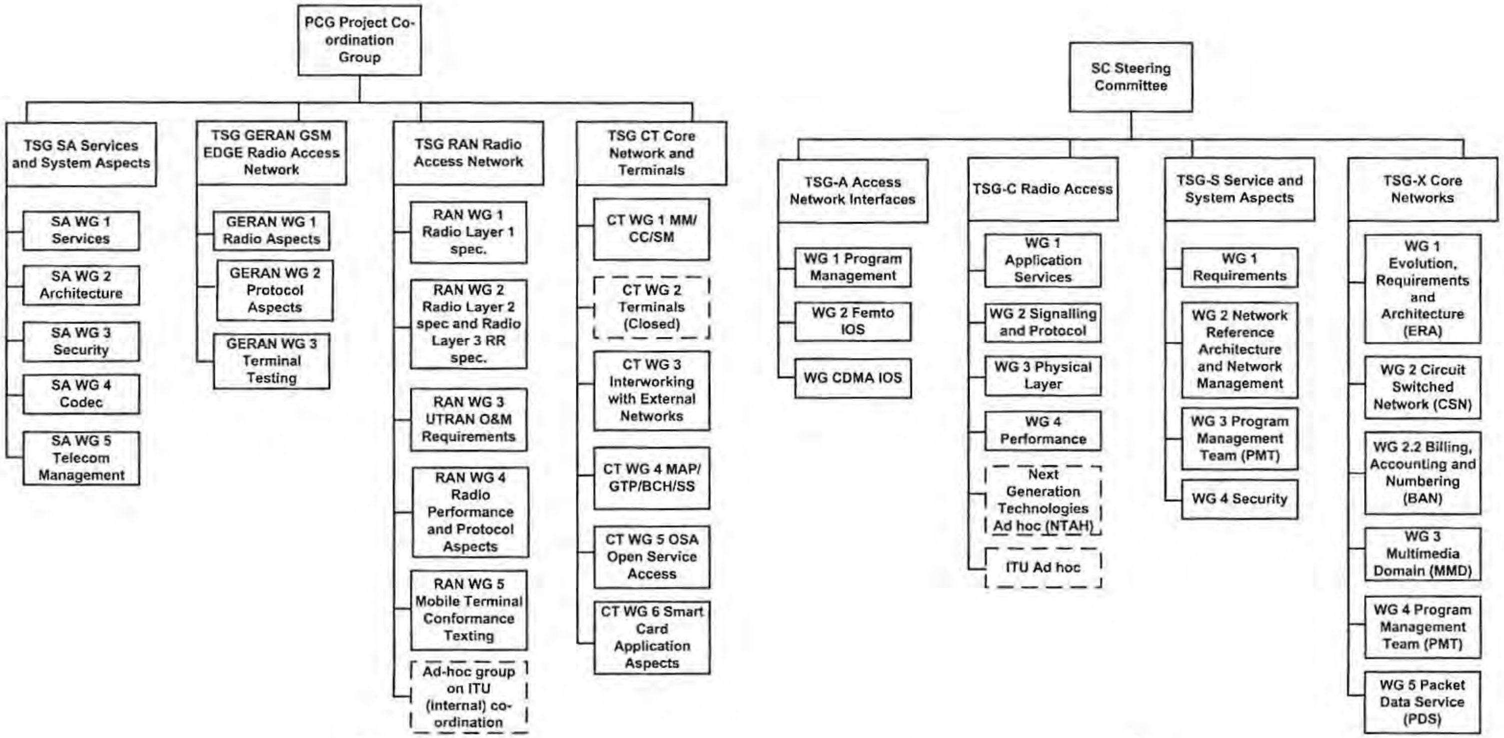


Figure 1.1: The organisational structure of ETSI. Adapted from [1].

such as UTRA (Universal Terrestrial Radio Access)[24]. The original scope was later extended to include the maintenance of GSM standards and evolved packet-switched access technologies such as GPRS and EDGE. 3GPP consists of a number of regional standardisation bodies, which are known as organisational partners, among which are ARIB (Association of Radio Industries and Business) and TTC (Telecommunication Technology Committee) in Japan, ATIS (Alliance for Telecommunication Industry Solutions) in America, ETSI. 3GPP is organised into a Project Co-ordination Group (PCG) and Technical Specification Groups (TSGs), whereby the PCG is responsible for the high-level management of 3GPP and the TSGs do all the core technical work [9].

1.2.1.3 3GPP2

3GPP2 is the sister organisation to the 3GPP and was formed after discussions between ETSI and some special interests groups from North America and Asian countries [25]. The dialog was based on the identification of the need to create another partnership project that was similar in nature to the 3GPP but would concern itself with the evolution of CDMA2000 (Code Division Multiple Access 2000) and other related ANSI standards. Many of the organisational partners that are involved in 3GPP such as ARIB and TTA are also involved in 3GPP2. The structure of 3GPP2 is similar to 3GPP, and consists of working groups which are grouped into TSGs. Figures 1.2a and 1.2b show the structure of the 3GPP and the 3GPP2 respectively.



(a) (b)

Figure 1.2: The organisational structure of (a) 3GPP and (b) 3GPP2

1.2.2 Related Standardisation Bodies

1.2.2.1 IETF

The IETF is the primary driving force behind the evolution of the Internet and Internet standards. It is an open, international community consisting of network designers, operators, vendors and researchers who are concerned with the development of Internet-based architectures and the smooth running of the Internet [26]. The IETF is responsible for developing technologies such as DNS (Domain Name System) and HTTP (Hyper Text Transfer Protocol). Unlike other SDOs, it is not a corporation and as such does not have a board of directors or corporate hierarchy, but is open to any interested individual without fees or membership dues [9].

The IETF uses the concept of working groups to accomplish technical work. Working groups are organised into Area Directorates, each of which is assigned a director. There are eight such directorates. Administrative oversight in the IETF is provided by the IESG (Internet Engineering Steering Group) which consists of one or two serving directors and the IETF chairman. Internet standards begin as Internet Drafts and eventually mature to become RFCs (Request for Comments).

IETF's collaboration with the 3GPP and 3GPP2 is documented in RFC 3113 [27] and RFC 3131 [28]. The guiding principle is such that SDOs will use IETF protocols where required, but when IETF protocols do not meet the requirements, they will submit such requirements to the IETF for development. Officially, this collaboration is driven by liaison officers in the respective camps, but the actual technical work is carried out through mailing list discussions, face to face meetings and special workshops. The result has been the development and extension of IETF protocols such as SIP (Session Initiation Protocol) in the form of Internet Drafts and RFCs that are designed to meet the needs of IMS. 3GPP requirements for extensions to SIP are documented in RFC 4083 [29]. Due to the sheer amount of these requirements, some of them are being addressed in the main SIP working group while others have been distributed among other working groups such as MMUSIC, SIMPLE and ROHC. The SIPPING (SIP Project INvestiGation) working group was formed to organise, document and prioritise requirements for SIP and forward them to the relevant IETF working group.

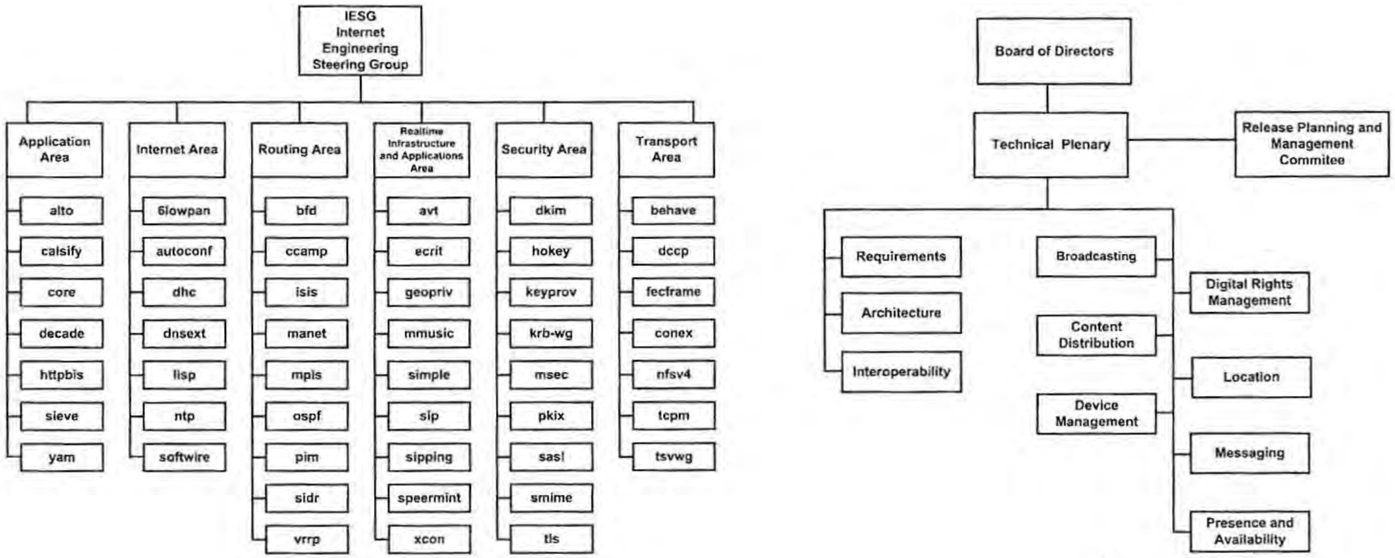
1.2.2.2 OMA

The OMA is the leading industry forum for the development of market-oriented, interoperable, mobile data services [30]. OMA was formed in 2002 and is the result of the amalgamation of previously existing industry fora such as the Wireless Village, the Wireless Application Protocol (WAP) Forum, the Mobile Gaming Interoperability Forum (MGIF), the Location Interoperability Forum (LIF) and the Mobile Wireless Internet Forum (MWIF). There are almost 200 member companies in OMA today from various sectors of industry such as mobile operators, device and network suppliers, as well as content and service providers. The technical term for a data service developed by OMA is a service enabler, of which OMA has developed over 100 in areas such as device management, Push to Talk over Cellular (PoC), presence, instant messaging and location services. At the heart of the OMA structure is the Technical Plenary which consists of the OMA working groups and two committees, the Operations and Processes Committee and the Release Planning and Management Committee.

There are OMA service enablers that use IMS as their base. An example is the OMA PoC enabler which is designed to provide a walkie-talkie-like function at the touch of a button to an operator or friends in a user's contact list. PoC uses SIP as its signaling protocol. In addition, services like PoC stand to gain much from existing infrastructure in the IMS. These and other dynamics of the relationship between OMA and the SDOs are described in OMA's technical specification on the utilisation of IMS capabilities by service enablers [31]. Generally, OMA is responsible for defining the requirements of service enablers and the SDOs must factor in those requirements into their specifications. This ensures that the OMA does not create a different version of the IMS for its enablers. Where overlaps do exist, OMA and the SDOs aim to have complementary specifications. This is the case, for example, with presence and messaging services. To ensure that OMA enablers are developed in an interoperable way, OMA developed an "IMS in OMA" enabler that highlights the healthy co-existence of service enablers in an IMS environment [32]. The structures of the IETF and the OMA are depicted in Figure 1.3, though many of the existing and active working groups in the IETF have been omitted for simplicity.

1.2.3 The Development of Technical Specifications

The various standards development organisations have their own internal processes for evolving proposals into technical specifications. As an example, Figure 1.4 shows how the process is achieved in the 3GPP. The procedures are roughly similar in the 3GPP2 as well. The process is centred around work items, which are requests for new network features. Work items are sometimes small in nature and in such cases, they can be handled by a single working group. In other cases, the items are large and can cut across multiple working groups. Measures are usually taken to avoid the overloading of individual working groups with too many work items,



(a) (b)

Figure 1.3: The organisational structure of (a) the IETF and (b) the OMA.

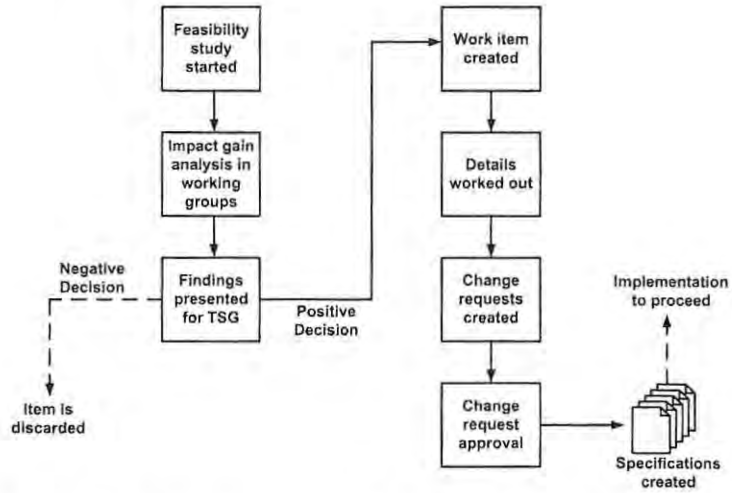


Figure 1.4: The process of the development of technical specifications in 3GPP. Source: [2]

especially when those work items are large. A work item must have the backing of about four companies and in certain cases, a feasibility study must also be conducted before an item can be entertained at working group level. The feasibility study is a way of measuring the cost versus benefit of introducing the feature into the existing architecture [33]. The passage of a work item to a working group occurs only at the discretion of the relevant TSG.

For each work item, an individual known as a rapporteur is nominated by the TSG to represent the working group to the TSG. The job of the rapporteur is to report on the progress of work items. Regular plenary meetings of the TSGs occur during the year to evaluate the relationship between the working groups and the TSGs. The progress of work items is evaluated at these meetings. This evaluation may lead to a work item being altered or even removed from the working group program. When a work item has been completed, a request known as a Change Request is brought before the plenary for its approval. The TSGs develop, approve and maintain technical specifications and technical reports through the working groups. Once a document has been approved by the TSG, it is submitted to the organisational partners to undergo standardisation. Thus, organisations such as the 3GPP and 3GPP2 develop technical specifications, whereas the organisational parters develop them into standards.

Technical documents in both the 3GPP and 3GPP2 follow strict numbering schemes. In the 3GPP, the numbering scheme follows a `X.YversionZ` pattern, where the `X` refers to the release number (releases are discussed in section 1.3), the `Y` refers to the version number and the `Z` refers to the sub-version number. For example, the document numbered TS 23.002v8.4.0 refers to release 8, version 4 and sub-version 0 of the technical specification numbered 23.002, which is 3GPP's technical specification on 3G Network Architecture. The 3GPP2 numbering scheme is slightly more complicated and follows an elaborate `A.Bcccc[-ddd]-Xversiony.z` pattern. The `A` represents the TSG behind the development of the document and the `B` represents the document type, which can be project, report or specification. The `cccc` is merely a sequential number for the document. The optional `dddd` part is for multi-part documents. The letter `X` denotes the revision of the document, which begins at 0 for the initial release, `A` for the first revision and so on. As an example, the document numbered X.S0013-002-B version 1.0 denotes revision B of the technical specification on the IP Multimedia Subsystem (Stage 2) under the TSG on Core Networks.

1.3 The Evolution of IMS

Every technical specification or report has in its identifier the corresponding IMS release number for it. Releases are iterations in 3GPP specifications that allow for a consistent reference point for anyone describing the IMS or parts of it. Originally, releases were meant to be published on a yearly basis, after which the release would be frozen. The act of freezing a release means that a consensus has been reached on the scope of that release and no additional features would be added to it. In practice, the 3GPP employs a strategy of staging its releases so that a release can contain a number of frozen stages, a system that was borrowed from the ITU [34]. Stage 1 focuses on describing a service from a user or operator's perspective. Stage 2 discusses network functions and maps service requirements to the capabilities of the network. The last stage, stage 3, covers the protocol signaling capabilities that are necessary for supporting the services that are described in stage 1. Table 1.1 provides a breakdown of past, current and anticipated 3GPP releases and the dates in which they were frozen or are expected to be frozen [1].

Common IMS refers to the version of IMS that is common to all SDOs, and was only realised in 3GPP Rel-8. It arose initially from the inclusion of both fixed-line and cable network access to an IMS that was dominated by radio access technologies. Common IMS was strengthened over time through collaboration between 3GPP, TISPAN and the cable network industry consortium known as CableLabs. Some of the work items regarding IMS that were under study in ETSI TISPAN were relocated to the 3GPP. Additionally, IP Multimedia content in CableLabs' specifications

[1] Some releases such as Rel-6 do not have an exact date but a duration for the freezing time. This is because OA&M (Operations, Administration and Maintenance) procedures in the development process can cause freezing to lag for some time. Rel-11 freeze dates are succeeded by question marks because these dates are those that were or are expected.

Release	Specification version number	Functional freeze date
Rel-11	11.x.y	Stage 1 freeze September 2011?
		Stage 2 freeze March 2012?
		Stage 3 freeze September 2012?
Rel-10	10.x.y	Stage 1 freeze March 2010
		Stage 2 September 2010
		Stage 3 March 2011
Rel-9	9.x.y	Stage 1 freeze December 2008
		Stage 2 June 2009
		Stage 3 freeze December 2009
Rel-8	8.x.y	Stage 1 freeze March 2008
		Stage 2 freeze June 2008
		Stage 3 freeze December 2008
Rel-7	7.x.y	Stage 1 freeze September 2005
		Stage 2 freeze September 2006
		Stage 3 freeze December 2007
Rel-6	6.x.y	December 2004 - March 2005
Rel-5	5.x.y	March - June 2002
Rel-4	4.x.y	March 2001
Rel-99	3.x.y	March 2000

Table 1.1: 3GPP freeze dates. Source: [14]

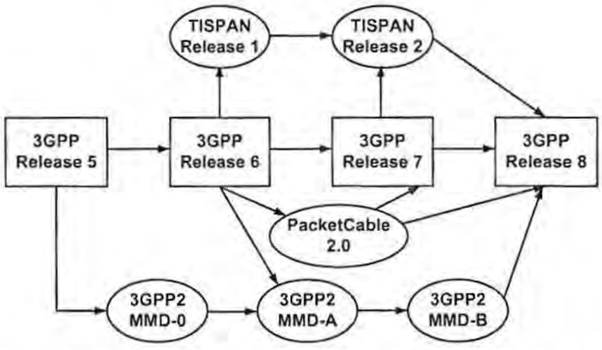


Figure 1.5: The evolution of IMS releases towards Common IMS. Source: [3]

were absorbed into 3GPP Rel-7, with the finalisation of the entire process reaching completion in Rel-8. Figure 1.5 summarises the evolution of releases towards Rel-8 which specifies Common IMS.

1.4 Services in IMS

The IMS has a highly structured service architecture. This section describes how services are hosted, the mechanisms that have been put into place to enable their execution and provides an example of how a typical service in IMS is delivered to the user.

1.4.1 The IMS Core

The core part of the IMS consists of three call session control functions (CSCFs) and a master database known as the Home Subscriber Server (HSS). The CSCFs are SIP servers that provide and support functions such as SIP message routing, registration, authentication and authorisation. The HSS is a master database that contains network data as well as user profile information. The IMS Core entities play an important role in service delivery since the user does not have direct access to the service layer[2], but all messages must traverse the IMS Core before arriving at the service layer.

[2]Except for service configuration

1.4.2 Application Servers

Services in the IMS are hosted on special servers known as application servers (ASs), much like web sites are hosted on web servers. There are three different types of application servers that have been defined. One of them is the SIP application server. It hosts SIP-based services and is typically the only one of the three types of application servers in which new services developed exclusively for IMS are found [9]. The others, the OSA SCS (Open Service Access Service Capability Server) and the IM-SSF (IP Multimedia Service Switching Function) are essentially gateway servers that reside in the home network and allow access to services developed in other environments. It is also possible to import SIP-based services hosted from a third party domain into the network.

The OSA SCS uses an OSA interface to interact with an OSA application server which implements services in a third party domain. OSA defines an architecture that allows developers to create new services that make use of network functionality that is exposed by the network. Typically, these services are developed using tools such as the OSA API and Web Services [35]. The IM-SSF is a gateway server towards legacy 2G mobile systems. It uses an interface called CAP (CAMEL Application Part) to interact with the gsmSCF which is part of the legacy service environment called CAMEL (Customized Applications for Mobile network Enhanced Logic). All IMS application servers implement two interfaces towards the IMS Core. One is the ISC (IP Multimedia Service Control) which is a SIP interface and the other is the Sh interface which is based on Diameter. Diameter is an IETF protocol that is used in AAA (Authentication, Authorisation and Accounting) applications in IP networks [36]. Figure 1.6 provides an overview of the IMS service environment, showing the standard application servers and the interfaces they implement.

1.4.3 Service Execution

In certain cases, an application server can execute a service without the user explicitly initiating a request for it. For example, subscription services that are activated after a certain time interval exhibit this behaviour. In most cases however, service execution begins with an explicit request from the user through their device, also called user equipment (UE). The UE either discovers during boot-up or is pre-programmed with the network location of the CSCF that is responsible for handling its request. When the request is received, a special CSCF in the IMS Core known as the serving CSCF (S-CSCF) must then decide how to handle it. If the request is for a service to be

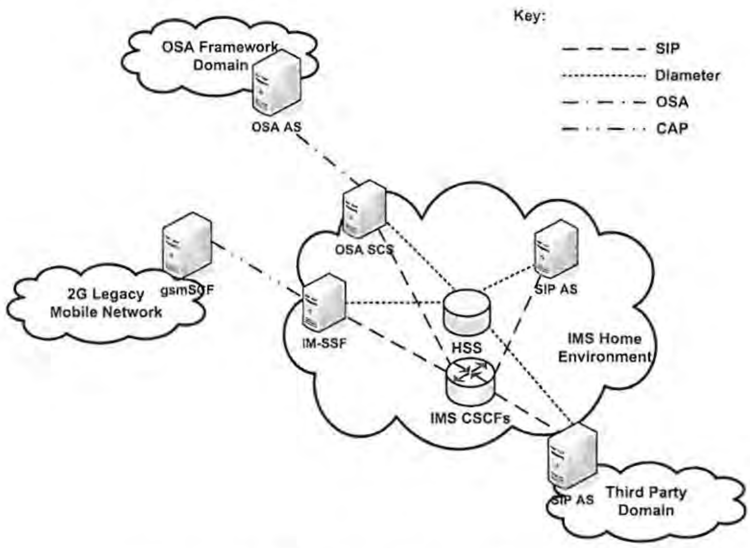


Figure 1.6: IMS application Servers.

executed (as opposed to a voice or video call to be initiated, for example), then the S-CSCF must pass the request to the application server that is responsible for executing the requested service. The user identity is contained in the SIP request it receives, so it extracts this information and looks up the rules that dictate where to route the request to next. If those rules are not immediately available to the S-CSCF, it must download them from the HSS which is responsible for storing this information in the user's profile.

Each user profile contains a special set of rules called iFC (Initial Filter Criteria) that contain information regarding the properties that a SIP request must exhibit in order for it to be delivered to an application server. In certain cases, a request can match more than one iFC instance, however because each iFC instance has a priority assigned to it, the request is routed to all application servers sequentially in order of priority. At each step, the indicated application server performs application logic that delivers some value added service to the user, such as delivering a video channel, initiating a call forwarding service, sending an email or updating buddies on the user's current presence status.

As an example to illustrate this, let the reader consider a multimedia service that merges web and telecommunication features to create a converged multimedia application. When the user logs on, the system establishes an association between the username specified in the web page and the IMS user identity linked to that username. Once this is done, the user clicks on a link in the web page and requests a video stream to be streamed to their UE. This prompts the application server to initiate a SIP call with the user. The call traverses the IMS Core and is routed by the

S-CSCF to the registered user's device. During the call, the system plays an announcement and the user confirms that they would like to receive the streamed video by pressing the call button. This generates a SIP re-INVITE message that is sent towards the application server by matching the SIP request against stored iFC. The application server then streams the video to the user. Should the user be interested in ordering the full video on DVD, she would be able to place an order through the web site. To ensure that the user's account has not been hacked, the system places a call to the user. Another announcement is played back and the user confirms the order by pressing a key in their handset. The key press generates a SIP OPTIONS request which is also matched against iFC and is routed to the application server which finalises the order and terminates the call.

1.4.4 Challenges and Opportunities

The service model described above is referred to as static service interaction management. Interaction management is the coordination of the interactions between service units that are required to deliver a complete service offering to a user [37]. This activity is static because it relies on the correct matching of rules that are hard-coded into the user profile, which rarely change. Static service interaction management works well in most cases, however it faces challenges in the wake of new developments in service creation in IMS. Worse yet, it fails to capitalise on certain opportunities that would result in significant advantages to both the operator and the operator's subscriber base. These issues are discussed next.

1.4.4.1 Service Decomposition and Service Interactions

The converged video streaming service previously described is a complex service with four different components. There is: a component that interacts with the web server, a component that handles the call state with the user, a component that provides DTMF event handling and finally a component that interacts with the streaming media server. In experimental and real world scenarios, an application server would house most if not all of these functions in a single network host.

There is an emerging trend in service development in IMS that encourages operators to develop services from the basic building blocks that are needed to realise them [5][4]. This strategy could significantly reduce code redundancies in large systems, would allow the re-use of functions across several servers and would enable rapid development of complex services. Some of these building blocks have been identified and standardised, and are referred to as service capabilities. The standardisation of service capabilities is seen as an important move to guarantee that there is a standard way of creating services. This would guarantee at least a minimal number of services in 3G networks and would also support service level interworking between operators implementing similar service capabilities. The standardisation of service capabilities instead of actual services themselves ensures that operators are still able to differentiate themselves through innovative service offerings.

While this movement is clearly beneficial, it challenges the current model of static service interaction management. This change means that the IMS Core would need to manage the interactions between several service capabilities deployed not only in different application servers, but potentially in different types of application servers. Due to the co-operation between IMS and OMA standards, there is an additional need to manage the interaction between service capabilities, OMA enablers and existing IMS services as well. Also, a single application server hosting all of the service capabilities needed for a single service is able to handle conflicts between the components it has control over. However, when service capabilities, OMA enablers and IMS services are distributed across several nodes, it is more difficult to handle and monitor potential conflicts and to avoid interactions that are undesirable or could cause error conditions.

1.4.4.2 Service Personalisation and Operator Policies

Static service interaction management means that the interactions between service units are governed by a set of conditions that are static in nature. Because SIP is a text-based protocol that uses header and value pairs to communicate the properties of a request, the only conditions that can be used to interrogate a request to see if it complies with iFC are SIP-based properties such as the SIP method used (i.e. INVITE), the indicated Request-URI (Uniform Resource Identifier) such as sip:bob@ims.com, the Session Case (i.e. Terminating) and others.

There are many scenarios beyond those that can be captured using the SIP protocol that might be appealing to operators in their efforts to add more value to the services they are offering to their subscribers. For instance, consider the converged video streaming service. On their birthday, users might enjoy the benefit of a short birthday video that automatically plays for a few seconds just before the actual video is delivered. That video could feature a celebrity or icon. This is a way of influencing the delivery of the service using the date as a determinant. Or perhaps, users might like the system to stream at one quality during working hours where they have access to a fast wireless connection, but at a lower quality after hours when they are at home on a slower connection. This is a way of influencing the delivery of a service using the time of day as a determinant. The operator may also wish to offer differentiated marketing advertisements in the videos streamed to its subscribers based on the manufacturer of the user's handset or the capabilities of that handset. This is a way of influencing the delivery of a service using a condition that could only be determined at runtime. The ability to cater for these kinds of situations would enable a dynamic form of service interaction as opposed to a static one. Dynamic service interaction management is a topic that has been addressed in the TR 23.810 technical report[3] published by the 3GPP, but as yet, there are no standards that detail how this process should be done.

1.4.4.3 Service Brokering and Policy Management

In addition to dynamic service interaction management and the specification of personalised service execution, there is another class of features that would provide an interesting set of benefits. The existence of service capabilities can be advertised using a secure interface in order to allow third parties to discover the capabilities of the network. This would require the use of a service broker that is responsible for brokering on behalf of third parties who use service capabilities to create services in their domains. Functions such as the authentication and authorisation of service requests and the discovery and secure access of service capabilities would also need to be performed. These functions could be assisted by service agreements between operators and third parties that govern the nature of the relationship.

1.5 Objectives

The primary objective of this thesis is to investigate the important yet under-researched topic of complex service interaction management. This investigation will describe what the 3GPP has contributed to the topic, list the currently unsolved problems and summarise the subsequent efforts of the research community to address these challenges. The investigation will present the most prevalent theories and critically analyse the solutions that have been proposed by the research community. This will ultimately lead to the presentation of a set of features and functions that have been deemed necessary for enabling complex service interaction management. These features and functions will be collected in a concise manner and used to formulate a novel architectural design.

[3]The content of this technical report is summarised in Chapter 2

The second objective is to evaluate the suitability of currently available and popularly used open source software for implementing an IMS testbed that accords with the service principles that have been identified. The aim is to assess the possibility and ease with which a researcher can utilise these platforms to deploy IMS services. Challenges such as the implications of working with a testbed that comprises heterogeneous components must be addressed in comparison with the utilisation of a single, common development platform.

The third objective is to design, implement and evaluate an initial prototype for the extended version of the IMS service layer that features a SCIM and includes additional features that assist the SCIM in performing its duties.

1.6 Scope

Ideally, dynamic service interaction management would include the handling of interaction between service capabilities in all three types of application servers. However, the scope of this investigation is limited to interactions involving only SIP-based service capabilities. The objective of this restriction is to simplify the solution architecture and focus on the core set of problems that are introduced by the new requirements, which can be addressed sufficiently within the scope of this category of service capabilities alone. That said, since the OSA SCS and the IM-SSF are no different from SIP application servers from the perspective of the IMS Core, any conclusions derived from the investigation should in theory apply to these environments as well.

It is important to differentiate between a service capability interaction manager and a service broker. The former performs the function of interaction management, while the latter is a more complex entity. This thesis maintains that an interaction manager is a subset of a service broker since the service broker performs interaction management in addition to other functions as well. Due to the complexity of incorporating the interaction manager alone into the IMS Core, the investigation and implementation of the service broker is outside the scope of this thesis. That said, developing a service capability interaction manager assists in providing a significant part of the feature set of the service broker and can be used as starting point for further investigation into service brokering in IMS.

1.7 Research By-products

The by-products that emanated from the thesis are:

1. The creation of an important bridge between the incomplete IMS technical specifications on service capability interaction management and the current state of the art (as reflected by research outputs of the research community) through the critical analysis of current material. This analysis also distinguishes between the terms SCIM and service broker where some researchers have used the two interchangeably.

2. The co-development of a modified version of the open source Mobicents Converged Demo with video trailer streaming. This system was made available to the Mobicents online community, with complete source code and ready-to-run JBOSS application server. An online tutorial of the system was also posted on YouTube and was well received by the Mobicents community. This is evidenced by the number of views it has registered, which is over 600 at the time of writing.

1.8 Document Overview

This thesis consists of seven chapters and five appendices. It is organised as follows:

Chapter 2 provides a rigorous review of the main technical specifications in the area of IMS that cover aspects related to service design and execution. Key concepts that are important for understanding the work described in this thesis such as service capabilities, interaction management and service personalisation are defined. The main entities that comprise the IMS service layer are presented and their protocol interfaces and functions are described in full. Examples of standardised services are described before the service capability interaction manager is introduced briefly and will be revisited in greater depth in the next chapter.

Chapter 3 builds on the discourse in the previous chapter on services and focuses specifically on the topic of the impact of introducing the service capability interaction manager into the IMS. The chapter begins with a brief summary of the 3GPP technical report on service capability interaction management which symbolises the 3GPP's first and only technical specification that covers this topic in depth. From here, the most significant research outputs authored by members of the research community on the topic of the service capability interaction management are reviewed and a critical analysis of their proposals is carried out. The chapter asserts that there are four main topics of concern regarding the service capability interaction manager, namely architectural placement, algorithms for interaction management, feature interactions and service personalisation. A brief section towards the end of the chapter is dedicated to an expanded view of the function of the service capability interaction manager and recognises that it is actually a component of a more complex network node, which is the service broker. The topic of service brokering, while interesting and pertinent to the topic of service capability interaction management, is however outside the scope of this thesis.

Chapter 4 presents an alternative design of the IMS service layer that is better equipped to cater for the requirements that are identified in Chapter 3. As it is an extended design that is able to handle both the current service model and the new requirements, it is given the name Extended IMS Service Layer, or EISL. The design is constrained by two requirements which are that, firstly, it must minimise the overall effect of the interaction manager on the existing service architecture, and secondly, it must introduce as few changes as possible to the behaviours and interfaces of existing IMS entities. A discussion follows that shows how the design fulfills all the requirements and ends with an explanation of how subscribers can use the existing functionality on their IMS terminals to personalise their services. EISL is a high-level architectural specification that is divorced from any implementational biases.

Chapter 5 represents the first of two stages towards an implementation. In this chapter, the practicalities associated with the development and deployment of services in an IMS testbed are explored and a summary of the most popular and widely used software platforms is provided. The design of an IMS service layer is presented that comprises several different parts. An alternative model is presented in the second half of this chapter that is based on the Mobicents communication suite. As a precursor to the implementation of the interaction manager, the implementation of a video streaming application is also presented.

Chapter 6 relates the second stage of the implementation process that details the use of open source software to implement EISL. In particular, it provides an initial proof of concept implementation of a SCIM which was developed using the SIP Servlet container of the Mobicents communications suite. The chapter begins with a justification of the use of SIP Servlets instead of the JAIN SLEE server since the two are competing technologies for the development of SIP application services. The system is described in detail and a discussion is provided that describes how the interactions of the EISL components with the open source IMS system were configured. Lastly, an example of service construction in a decomposed service environment is given.

Chapter 7 presents the key conclusions of this thesis. The limitations, shortcomings of the design and areas for future work are discussed.

Chapter 2

Service Theory and the IMS Service Architecture

There is a considerable amount of effort that has been invested and continues to be invested in the development of technical specifications for IMS. Some of these specifications eventually produce industry standards. Once standards exist, it is then possible for operators to work hand-in-hand with equipment vendors in the installation of the IMS network. The capital investment required for such projects must be appropriately justified and the operator must be confident that it will receive a satisfactory return on that investment. Telecommunication services have an important role to play in achieving this objective, as the services that an operator offers to its subscribers form an integral part of its revenue stream. As such, a significant amount of attention has been paid to the formalisation of service concepts in IMS. A taxonomy of IMS services has been developed that defines the different types of services that exist and how they are related. In order to diversify their product offerings, operators should be able to provide the full spectrum of services — from simple services that are cheap to acquire, to complex ones for higher income consumers. To address this need, technical specifications define a set of service building blocks known as service capabilities, each of which provides a single function. A simple service may consist of one or two service capabilities, whereas a complex service consists of multiple service capabilities. There is also a standard architecture for the service layer that is responsible for serving user requests and executing services. This chapter addresses the topic of IMS services, and in particular, shows that the term service is a highly overused term, and in fact exist in many forms. In truth, they exist in many forms. The IMS service layer is presented and the procedures and mechanisms involved in service execution are described in detail. At the end of the chapter, concerns about the current service architecture are highlighted as a precursor to the next chapter, which deals more robustly with advanced topics in service delivery.

2.1 Services and Service Concepts

Figure 1.2a shows that under the Services and System Aspects TSG of the 3GPP, there is a working group called SA WG1, also known as SA1. This working group sets high level requirements for the 3G system, and does so through Stage 1 (see Section 1.3) descriptions in the form of technical specifications [38]. Among the main focus points of SA1 are the following:

- Specification of features
- Specification of services
- Specification of service capabilities
- Identification of requirements to support service creation
- Identification of requirements for service interworking
- Identification of requirements for service interoperability between networks

The objective of this section is to define the terms above and to describe how service creation is achieved by recycling underlying functionality.

2.1.1 Features and Building Blocks

In Section 1.2.3, the process of developing technical specifications was discussed and it explained that new features could be added to the system once a work item is complete and is submitted as a Change Request to the TSG. There are four kinds of work items: a study item, a feature, a building block and a work task [39]. A study item is defined as a type of work item that conducts a feasibility study that results in the publication of a technical report. Once the technical report is completed and has been accepted, new features are defined which could potentially be added to the existing system. A feature is a new or enhanced functionality which adds value to the existing system and usually embodies an improved service to the customer or increased revenue generation potential to the supplier.

A building block is a division of a feature that represents a self-sustained technical functionality which is typically resident in a single network element. Features can be divided into building blocks if they are sufficiently complex in order to allow for more precise definitions. By nature, a building block may be re-usable, and as such, may be common to more than one feature. A work task is a division of a building block and represents a self-contained and scoped item of work.

2.1.2 Service Concepts

TS 22.105 describes the full spectrum of network services and defines a hierarchy of telecommunication services that classifies services based on their complexity [4]. The two most basic services are bearer services and teleservices. A bearer is simply an information communication path, while a bearer service is a type of service that provides the capability of transmitting signals along that path. A teleservice is a more comprehensive type of service that provides the complete capability, including terminal and equipment functions, for communication between users. An example of a teleservice is an emergency call, an SMS, a fax or a voice group call. Figure 2.1[1] shows the relationship between these service types. Both packet and circuit switched networks define bearer services, but only circuit switched networks define teleservices.

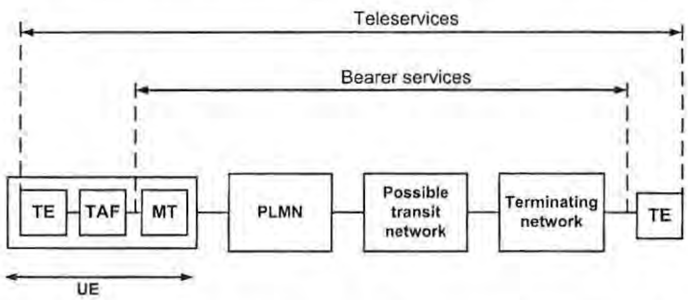


Figure 2.1: Bearer services and teleservices. Source: [4].

A third type of service is a supplementary service. A supplementary service modifies or supplements a basic telecommunication service such as a bearer service or a teleservice. Examples include call deflection, call transfer and call identification, which all supplement the basic voice or video call service. IP multimedia services are built from packet switched bearer services and are developed using SIP. Examples of these services include chat and shared whiteboard services. Other value added services that are not related to telephony are also provided for and include MMS (Multimedia Messaging Service) and Email. Toolkits such as CAMEL and OSA

[1] The components of the UE shown in this figure are discussed later in the chapter.

(discussed in Section 2.2.4) have also been defined by the 3GPP which can be used to create or modify the above types of services. A prepaid charging application is an example of a service that can be created using these kinds of toolkits. Figure 2.2 summarises these concepts.

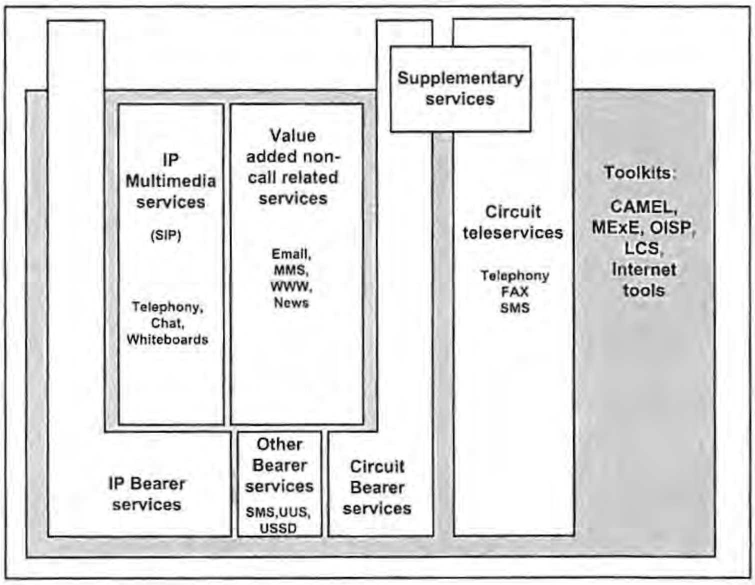


Figure 2.2: A classification of services. Source: [5].

TS 22.004 provides a general overview of issues related to supplementary services [40]. However, many of the concepts that are mentioned in this technical specification can apply to other service types as well. The document highlights the close relationship that exists between the network operator, the service provider, the service itself and the service subscriber. The technical specification describes several operations that can be performed on supplementary services either by the operator or the servider provider. Each of these operations can potentially affect the relationship between a service and a service subscriber.

Service provisioning describes the process that results in a service being made available to a subscriber. This provisioning can be global whereby these services are made available to all subscribers without active participation from them, or it can be pre-arranged, such as through the signing of a contract or by filling out a web form. Service registration is the process whereby either a subscriber or service provider provides information that enables the service to operate. Registration may also refer to the automatic activation of the service. Service activation is performed either by a service provider, a subscriber or automatically by the network, in order to enable the service to run as and when required.

Activation results in a service transitioning to a state known as the active state. Only when a service is in an active state can service invocation occur. Invocation is an action that is performed by a subscriber or by the network itself to invoke a required service. A subscriber can initiate the invocation of a service by the press of a button, for example. The network can invoke the same service by triggering a timer that results in an invocation request being made. Service withdrawal is performed in order to make a service unavailable to a subscriber. As with provisioning, the withdrawal can be global in scope or only affect certain subscribers. Service deactivation is an action performed by the service provider, the subscriber or the network that terminates the condition created by service activation. After a service is invoked, other services may need to become involved. Thus the 3GPP has identified the need for catering for interactions between services in order to allow simultaneous use of different services by a subscriber.

Erasure and interrogation are terms used to describe operations performed not on a service itself, but on information provisioned in the network related to that service or a subscriber. Erasure is performed by the service provider, the subscriber or the network, whereby information that was created through a previous registration is removed. Interrogation is usually performed by a subscriber and describes the action of the subscriber requesting information about a particular service. Valid forms of requests are status checks, data checks and data requests. A status check can return a value of "not supported", "active and operative", "active and quiescent" or "not active". A data check compares the data supplied by a subscriber to that which has been provisioned in the network, and can return either a value of "check is positive" or "check is negative". A data request allows the subscriber to obtain a confirmation of the data that was provisioned in the data check.

2.1.3 Service Capabilities

In its specifications, the 3GPP states that its policy is not to standardise the IP multimedia services themselves. The reason for this, according to [41], is that if the services themselves were standardised, it would inhibit the ability of operators to provide product differentiation and would also discourage innovation. Instead, the 3GPP standardises service capabilities.

Service capabilities are defined as the "*bearers that are defined by parameters and mechanisms needed to realise services*" [4]. This means that they are the basic capabilities that can be used to develop telecommunication services. Service capabilities can be made visible to other entities in the network such as application servers through an application interface. A service capability can provide more than one interface. This interface can be used to manipulate service capability features, which are the functions provided by the service capability itself. In this way, service capability features enable the use of the service capability in an open and secure way. In this thesis, for simplicity, the terms service capability and service capability feature are used interchangeably. Figure 2.3 shows the relationship between service capabilities and service capability features.

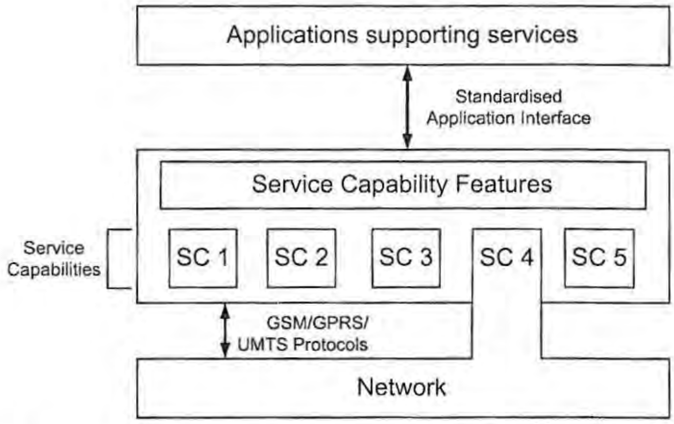


Figure 2.3: Service capabilities and service capability features. Source: [6]

The 3GPP defines two different types of service capability features: framework and non-framework. Framework service capability features provide commonly used utilities that are required for the non-framework capability features to be accessible, secure and manageable. This includes authentication, authorisation and notification. Non-framework capability features are those that will enable applications to make use of the underlying network capabilities. This includes session control, terminal capabilities and user location. A list of some of the service capability features that have been standardised is provided in Table 2.1. The list is derived from [5].

Framework Service Capabilities	**Non-framework Service Capabilities**
Authentication	Session Control
User-Network Authentication	Security/Privacy
Application-Network Authentication	Address Translation
Authorisation	Location
Application-Network Authorisation	User Status
User-Application Authorisation	Terminal Capabilities
Registration	Messaging
Discovery	Data Download
Notification	User Profile Management

Table 2.1: Examples of framework and non-framework service capabilities.

2.1.4 Service Creation

One of the chief benefits of service capabilities is that they enable rapid service creation. TISPAN has also identified this benefit and have incorporated the concept of service capabilities into their specifications. In TISPAN, the view is that an overall service consists of service applications. A service application is the actual technical functionality that is essential to the service that is delivered to the user. Service applications can be broken down into service capabilities, which by definition are re-usable so that multiple service applications can make use of the same service capability. TISPAN also sheds some light on the possible mechanisms and parameters that 3GPP associated with service capabilities by defining the following parameters which are found in [7]:

- A service capability must have an identifier or label
- A service capability must declare any attributes essential to it
- A service capability must define a set of normal behaviours essential to it
- A service capability must declare a set of behaviours pertinent to error conditions
- A service capability must define any interactions it has with other service capabilities

TISPAN has also detailed the advantages that this service creation approach provides to the various stakeholders. Service providers can combine service applications into new services that differentiate them from their competitors. Equipment manufacturers can create equipment that supports standardised service capabilities, adding their own customisations or extensions. Network operators can choose which network capabilities to offer in order to support a large set of services. Regulators may also require that network operators provide a minimum set of service capabilities to allow fair access to services, and to allow users to use their terminals with different service providers and on different networks. Figure 2.4 illustrates TISPAN's view of service creation through the inheritance of capabilities.

To further illustrate the potential benefit of this strategy, the 3GPP defines in [4] a set of supplementary services that could potentially be realised through such service compositions. These include:

Call Barring Call barring allows the prevention of certain outgoing calls based on conditions such as the number being dialed, and whether or not the callee is roaming. In IMS, it would be possible to perform barring of outgoing calls based on a wider range of parameters such

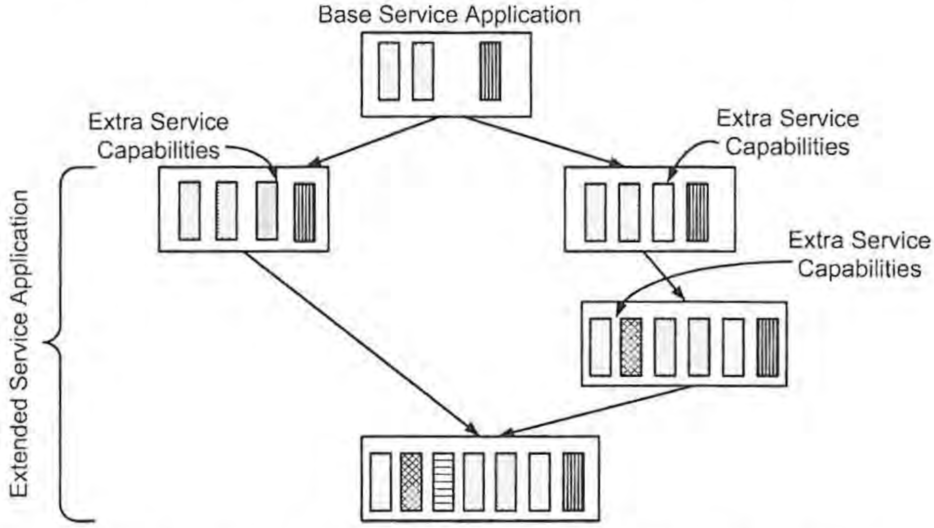


Figure 2.4: Service creation from service capabilities. Source: [7]

as the time of day, the day of the week, the location, the type of call requested or even the destination.

Call Filtering/Forwarding In the 2G mobile system, there is no support for call filtering. As such, all calls that can be terminated must necessarily be presented to the callee, except when the callee has activated the call forwarding feature. In addition, it is not possible to handle the forwarding of calls differently based on specified parameters. In IMS, the call filtering service could decide whether the call is to be accepted, forwarded or terminated, using parameters such as the caller ID, the time of day or day of the week, user profile settings or terminal status. The filtering function could also have two phases, one of which enforces immediate filtering irrespective of whether or not the callee is online, or late filtering which filters only when the callee is online.

Call Hold The call hold service permits the temporary maintenance of a call that has already been established, while suspending it from the incoming access point of the network. This suspension helps to conserve the radio interface and network resources.

Call Transfer The call transfer service allows a call that has either already been established or is in the held state to be transferred to another destination. This would be done by either establishing a new call towards the destination or transferring the existing call towards the destination.

Call Back When Free The call back when free service applies to voice or video calls as well as instant messages, and is invoked when the destination device is busy and the call or message cannot be delivered. The network can inform the initiator when the receiver is available again, and the call or message can be re-attempted. Furthermore, when there is an accumulation of undelivered requests, the service can decide in which order the requests are sent based on certain parameters.

2.1.5 Service Interworking

The service creation strategy described by TISPAN realises another benefit in addition to the re-use of service capabilities. The use of service capabilities fosters the definition of a common language that can be applied to any service. This means that service capabilities can be used to describe services in a similar way to how the letters of the alphabet are used to construct words and sentences. This is important because since service capabilities are standard, and services are described in terms of those service capabilities, it is then possible to support service-level interworking between different networks. Interworking can be achieved since the service capabilities can be individually mapped onto the individual network technologies in the different networks, such as different flavours of SIP.

Figure 2.5 shows service offering 1 and service offering 2. If the service offerings comprise of service applications A and B respectively, then the common set of functionality that can be delivered when these services interwork should be the common subset of service capabilities that belong to service applications A and B.

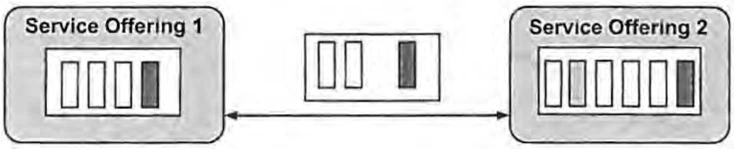


Figure 2.5: Service interworking through service capabilities. Source: [7]

2.2 IMS Service Architecture

The 3GPP technical specification TS 23.002 on Network Architecture provides an overview of the 3GPP Public Land and Mobile Network (PLMN). Included in this document is a detailed outline of the IMS, and in particular, the service architecture. The term service architecture refers to all the entities, configurations, interfaces and mechanisms in the IMS that are required for supporting multimedia services. The IMS service architecture is illustrated in Figure 2.6. Table 2.2 provides details of the protocol interfaces in the service architecture. There are three sections that can be identified in the service architecture which are the user domain, the IMS Core and the service plane.

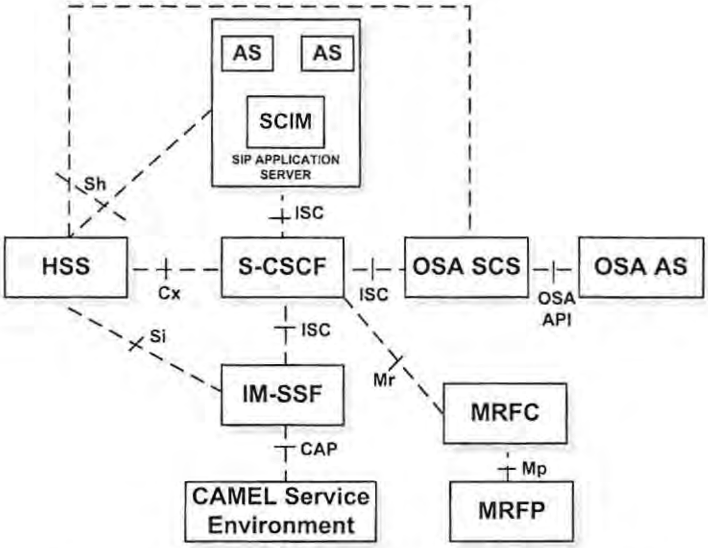


Figure 2.6: IMS Service Architecture. Source: [8].

2.2.1 User Domain

The user domain is where the user equipment (UE) resides. The UE is what allows a user to access the network, such as a mobile phone or tablet PC. The interface between the UE and the network is a radio interface. Furthermore, the UE is divided into the UICC (USIM IC Card) domain and the mobile equipment (ME) domain. The UICC is a physically secure device or an integrated circuit card (smart card) that can be removed from the terminal, and contains one or more applications on it. The ME domain is divided into the mobile termination (MT) and the terminal equipment (TE) components. The MT component supports functions specific to the management of the PLMN access interface, while the TE component provides the functions necessary for the operation of the access protocols by the user. On the application protocol level, the UE communicates with the IMS using the Gm interface which is implemented in SIP.

An IMS subscriber is assigned one or more IP Multimedia Private Identities (IMPIs), which are unique identifiers set by the operator and stored on the UICC. The IMPI is a non-SIP URI in the

Interface name	Protocol / API implementation	Meaning
Sh	Diameter	Used between the HSS and the application server. The application server uses it to download and upload user related information.
Cx	Diameter	Used between the HSS and both the I-CSCF (not shown in Figure 2.6) and the S-CSCF. It supports the transfer of information such as procedures related to S-CSCF assignment, routing information retrieval from HSS to I/S-CSCF and filter control.
CAP	CAP	Used between the IM-SSF and the CAMEL Service environment. It supports the re-use of services built in legacy mobile networks for IMS.
Si	MAP	Used between the HSS and the IM-SSF. It is analogous to the Sh interface.
ISC	SIP	Used between the S-CSCF and an application server (SIP AS, OSA SCS, IM-SSF). It is used to control services for IMS.
OSA API	OSA API	Used between the OSA SCS and the OSA AS. It allows the OSA AS to access network functionality through the OSA SCS.
Mr	SIP	Used between the S-CSCF and the MRFC. It allows the S-CSCF to trigger the MRFC.
Mp	H.248	Used between the MRFC and the MRFP. It provides the MRFC with control of media resources belonging to the MRF function.

Table 2.2: Protocol interfaces in the IMS Service Architecture.

form *user@ims.co.za* and is usually not known to the human user. An IMPI can be associated with one or more IP Multimedia Public Identities (IMPU), which are either SIP URIs or TEL (telephone) URIs of the form *sip:user@ims.co.za* and *tel:+2712345553313* respectively. The IMPU is known by the human user and can be distributed to the user's personal and business contacts.

The UE is capable of managing service configurations and service related information on behalf of the user. An example is when the user would like to set up a list of contacts that will be notified when their presence status changes. The UE achieves this via the Ut interface which supports the upload and download of such information and is implemented using a protocol that was borrowed from the IETF called XCAP (XML Configuration Access Protocol). XCAP is an HTTP-like protocol that uses request and response codes with XML payloads between a client and a server [42]. An XCAP server is referred to as an XML Document Management Server (XDMS). XCAP allows portions of a hosted XML file on the XDMS to be addressable using URIs that allow the modification of portions of that file. This granular addressing scheme is much more efficient than the alternative, which is to upload new versions of those files when they change, which would waste bandwidth. This mechanism is based on the XPATH protocol which was developed by the World Wide Web Consortium (W3C). Figure 2.7 shows an example of the retrieval of a buddy list (resource list) from the XDMS. In it, the user Bob uses an HTTP GET to request the list of buddies that he has inserted under a list called "Close-Friends". The XDMS responds with an HTTP 200 OK and attaches the list contents in an XML document.

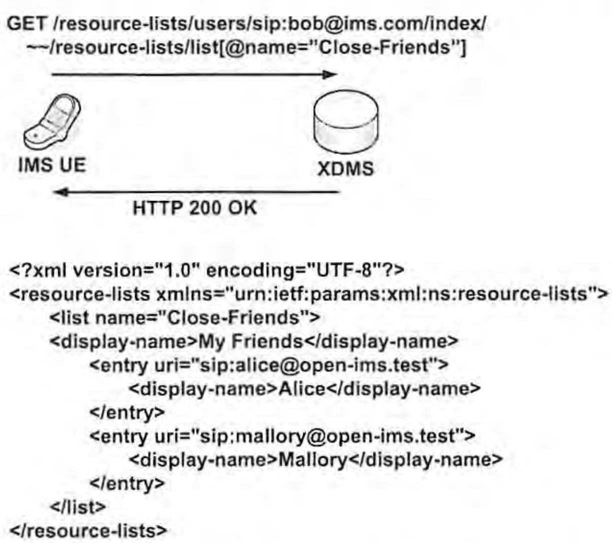


Figure 2.7: The management of service related information using XCAP.

XCAP defines use cases called application unique identifiers (AUIDs). Every AUID is associated with an XML schema that a file associated with it must adhere to. The schema defines rules for how applications are allowed to interact with resources stored in the hosted file. Examples of standard AUIDs are pres-rules (defines policies dictating what presence information can be revealed to watchers of a user's presence status), resource-lists (defines the management of contact lists) and rls-services (defines service URIs for linking groups of contacts).

2.2.2 IMS Core Network

The IP multimedia core network (CN) includes all the network entities that are required to support multimedia services [43]. For the purpose of this discussion, the focus is restricted to the call session control functions (CSCFs) and the HSS (Home Subscriber Server).

2.2.2.1 Call Session Control Functions

The CSCFs are SIP servers that satisfy the requirements outlined in the SIP standard [44], but also include many additional enhancements. There are three CSCFs, namely the Proxy, Interrogating and Serving CSCF. The IMS may have many instances of each type of CSCF for scalability and performance reasons. The Proxy CSCF, or P-CSCF, acts as both an inbound and outbound server for the UE, since all the SIP signaling between the UE and the network must traverse the P-CSCF. This occurs along the Gm interface. The P-CSCF performs several functions on behalf of the user. One of them is security. The P-CSCF uses IETF's IPSec protocol to offer integrity protection. The P-CSCF also authenticates the user in order to assert the user's identity, so that other nodes upstream do not also have to perform this duty, as the P-CSCF is a trusted node. The P-CSCF must also verify the correctness of the SIP messages that it receives so that when messages are passed upstream, they are guaranteed to be appropriately formatted. The P-CSCF decompresses messages from the UE and compresses those it passes on to the UE. Messages to and from the UE are subject to compression in order to reduce the size of messages that are transmitted over the air interface.

The Interrogating CSCF, or I-CSCF is the SIP server whose address is recorded in DNS for a given domain name. The I-CSCF maintains an interface known as the Cx interface with the HSS which is implemented using the Diameter protocol. It uses this interface to download user location information, which enables it to route an external request to the intended user. The I-CSCF also has an ISC (IP multimedia Service Control) interface which allows it to route external requests to an application server instead of a user.

The last CSCF is the Serving CSCF, or S-CSCF. It performs session control but also acts as a SIP registrar, which means it registers users onto the network. The S-CSCF, like the I-CSCF, has a Cx Diameter interface with the HSS through which it downloads information such as user profiles. The main function of the S-CSCF is to provide SIP routing functions. This includes the routing of requests that will lead to the execution of a service on behalf of the user, since it is the responsibility of the S-CSCF to transfer SIP requests toward the application servers that host multimedia services.

2.2.2.2 Home Subscriber Server

The HSS is a master database that is responsible for storing all permanent subscriber data and all relevant temporary subscriber data to support the call control and session management entities of the different domains and subsystems [45]. This information allows the HSS to provide support to the network entities that are responsible for handling session management, such as the CSCFs. Technically, there can be more than one HSS in the IMS network. In this case, a Subscription Locator Function (SLF) is used to map a given user to a corresponding HSS since the information related to a given user is never distributed across multiple databases.

Figure 2.8 shows the structure of the IMS user profile. It contains one or more service profiles, each of which is associated with at least one IMPU. The Core Network Service Authorisation part does not need to be specified, but can be used in certain cases to perform actions such as prohibiting an IMS user from using certain media formats. When present, the initial filter criteria (iFC) determine which application server the S-CSCF directs an incoming SIP request to. The decision is made based on the SIP-specific properties of that request. An iFC instance contains a priority parameter, which, depending on its value, determines the order in which the criteria are evaluated. The lower the number, the greater the priority.

An iFC instance is associated with two classes of information, a trigger point and an application server. A trigger point does not need to be specified, but when it is, it contains a single boolean parameter that determines whether the set of service point triggers it is associated with are expressed in a disjunctive or a conjunctive normal form. A service point trigger is a logical statement, expressed using SIP attributes. The classes of attributes that can be used are the Request-URI, the SIP Method, the SIP Header, the Session Case and the Session Description. Table 2.3 explains what these attributes mean and the kind of values each can take. A service point trigger must use at least one of these attributes, and assign to it an appropriate value. Each service point trigger can also be negated.

Conjunctive normal form (CNF) is a way of expressing service point triggers as an ANDed set of ORed statements. The following expression is in CNF:

$$[((Method = INVITE) \vee (Method = SUBSCRIBE)) \wedge !(Request - URI = sip : mosiuoa@ims.com) \wedge (SessionCase = Originating)]$$

This expression matches a SIP INVITE or SUBSCRIBE request that is originated by the user and the intended target's IMPU is not *sip:mosiuoa@ims.com*.

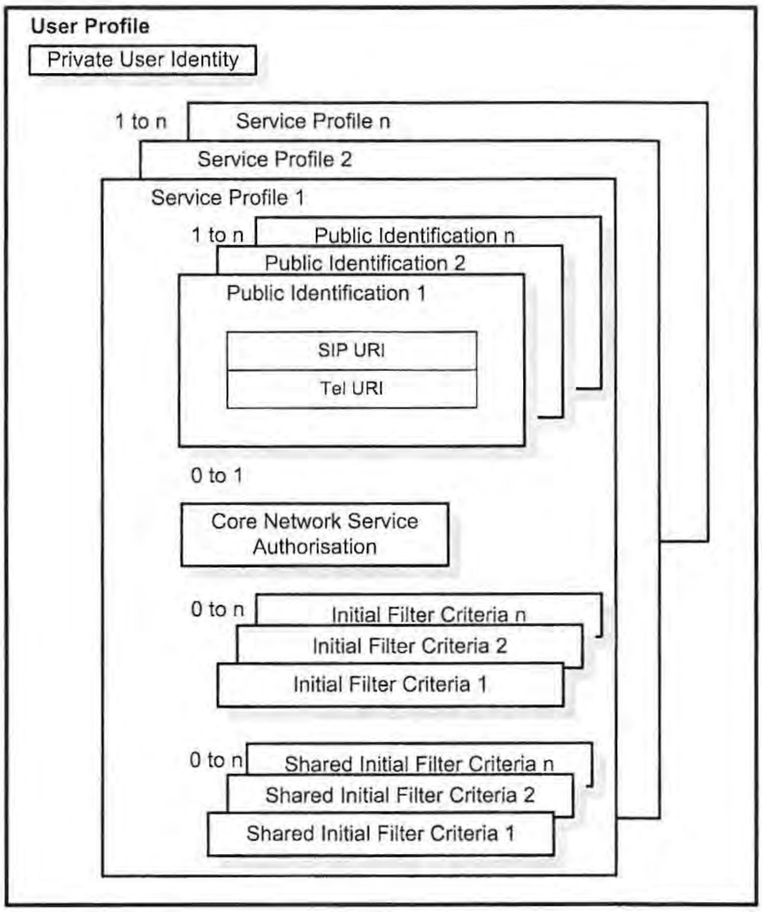


Figure 2.8: Structure of the IMS user profile. Source: [9]

Disjunctive normal form (DNF) is a way of expressing service point triggers as an ORed set of ANDed statements. The following expression is in DNF:

$$[((Method = MESSAGE) \wedge (ContentyType! = application/xml + pidf)) \vee (SessionCase = Originating)]$$

This expression matches a SIP MESSAGE request whose content type is not *application/pidf+xml* or any request that is originated by the user.

Trigger Name	**Properties**	**Description**	**Examples**
Request URI	Request-URI	The URI contained in the Request-URI header of the SIP method	sip:mosiuoa@ims.com
SIP Method	Method	The type of SIP request method	NOTIFY
SIP Header	Header Content	The type of SIP header The value of the SIP header	ContentType application/pidf+xml
Session Case	Session Case	Whether the request is originated by the user or terminated at the user, and indicates the state of the user	Originating, Terminating_Registered, Originating_Unregistered
Session Description	Line Content	Denotes a line in the SDP The value identified by the line	m:video 8000 RTP / AVP 34 21

Table 2.3: Service point trigger groups.

The application server part contains two parameters, of which one is its SIP URI, and the other is a parameter that determines the default handling of the request should the application server not respond timeously. The default method of handling will cause either the continuation of the SIP dialog or its termination. Besides these properties, an application server instance has three classes of information associated with it. The first is the service information class, which allows the S-CSCF to download the information that is sent to an application server when the iFC are satisfied. The second is a class of information that indicates to the S-CSCF that a SIP REGISTER request from the UE must be sent to the application server as a third party registration. The last class of information indicates to the S-CSCF that the final response to the REGISTER request must also be sent to the application server when the iFC are satisfied.

2.2.3 Media Resource Function

The main media function in the CN is provided by the Media Resource Function (MRF). It is distinct from the Media Gateway, which unlike the MRF, is located at the edge of the IMS domain and does media transcoding between the IP network and a circuit-switched network. The MRF is often decomposed into its two logical functions, which are the Media Resource Function Controller (MRFC) and the Media Resource Function Processor (MRFP). The MRFC and MRFP have a master-slave relationship. The MRFC interprets information coming from a requester, such as an application server or the S-CSCF, and controls the media resources in the MRFP to satisfy the request. The MRFC also generates call detail records (CDRs). The MRFP provides the resources that are controlled by the MRFC, and performs functions on media streams such as sourcing, mixing and processing of the streams, and can also provide floor control to manage media resources for a conference service.

2.2.4 Application Servers

An application server in IMS is a server that offers value-added IP multimedia services [46]. IMS defines three types of application servers, the SIP application server, the Open Service Access Service Capability Server (OSA SCS) and the IP Multimedia Service Switching Function (IM-SSF). The SIP application server hosts services that are developed using the SIP protocol. The OSA SCS appears as a SIP application server to the IMS Core, but implements an interface to an OSA application server in a third party OSA domain. OSA defines an architecture that allows developers of telecommunication services to utilise functionality native to the operator's network through an OSA API. The aim of OSA is to enable the development of future applications that are not known today by providing an interface that supports the manipulation of network features. The API allows developers to access features such as call control, messaging, mobility, terminal capabilities and billing, among others. The OSA API is independent of programming language, and currently there are Java SE (Standard Edition), Java EE (Enterprise Edition), and Parlay X (Web services) realisations of the API [35]. The IM-SSF provides a bridge between the IMS and the services which have been developed using CAMEL (Customized Applications for Mobile network Enhanced Logic) for the old 2G mobile system through the CAMEL Application Part (CAP) interface. CAMEL is a service environment that allows legacy mobile operators to create operator-specific services [47]. The IM-SSF translates requests between SIP and CAP.

2.2.5 Standardised Services

IMS is a docking station that provides a secure, quality of service aware and billable platform for service deployment. Interfaces such as CAP, OSA and SIP allow services to be attached to the IMS, even from outside the operator domain. These properties make it a powerful service delivery platform, providing IMS subscribers with ample services to choose from. In reality, there is a tendency for operators to make *walled gardens* of their networks, and to be hesitant to open them up to third parties. By restricting users to only those services that are offered in their home environment, the operator eventually starves subscribers of attractive services. At the same time, there are a plethora of free Internet services such as Facebook, YouTube and Twitter that provide Internet users with interesting services. Coupled with 3G data speeds, the Internet poses itself as a competing platform as subscribers will tend to take advantage of the carrier network capabilities in order to consume multimedia services directly from the Internet, leaving operators as nothing more than bit pipes [48].

In addition to standardising service capabilities as a way of showing operators the potential service delivery abilities of the IMS, the 3GPP has also focused on showing how services that are currently available on the Internet could be deployed in the IMS. For examples, technical specifications have been developed for common services such as presence, messaging and multimedia conferencing [49, 50, 51]. These technical specifications detail how these services behave, what the functional entities involved in delivering the services are and how these entities communicate with the current IMS network.

In some quarters, these services have been called service capabilities. Examples of this error are found in [52] and [53], but it is clear that these technical specifications describe whole services and not service capabilities, and complex ones at that. As an example, [49] describes a presence service in its totality, including functional entities such as a presence server, a presence network agent, an XCAP server, an event publication agent and an event state compositor. Entities such as XCAP servers are complex in their own right and cannot be regarded as basic building blocks. The specifications however, show how the service should behave and how it integrates with the IMS. This is not to be viewed as contradictory to their stance on service creation from building blocks, but should be viewed as an attempt to market IMS as a platform that can support revenue-generating services that are currently present in Internet environments.

The topic of whole services versus services composed from service capabilities was visited in the 3GPP mailing list of the SA WG1 [54] in February 2001. In this discussion, members of the working group acknowledged the importance of standardised service capabilities, and agreed that operators should deploy similar service capabilities to enable interworking. However, the challenge that they faced was that it would be difficult to guarantee that all operators provided similar services through the definition of service capabilities alone. It would also be challenging to guarantee that all operators deployed all necessary service capabilities, employing the same mechanisms, in order to enable a service. A solution that was suggested by members was that SA WG1 would choose a small set of services that could be of interest to operators and verify that service capabilities could in fact be used to fully enable these services. There does not seem to be any evidence that would suggest that this work was undertaken or was completed.

2.2.6 Service Execution in IMS

IMS possesses a well defined framework that governs how services are requested and delivered. Facets of this framework were covered in the discussion regarding iFCs and trigger points, however the complete picture requires an examination of the procedures at the S-CSCF and how it interacts with the HSS as well as the application servers. Because the interface between the S-CSCF and the application server (the ISC interface) is implemented in SIP, the procedures for all three IMS application servers are essentially the same from the perspective of the IMS Core. The OSA SCS can be regarded as a type of SIP application server that interfaces with the OSA framework, while the IM-SSF as a type of SIP application server that interfaces with legacy mobile systems.

The delivery of a service can be initiated either by a user or an application server. In the case of a request originated by the UE, the SIP request is received by the P-CSCF which performs the duties detailed in Section 2.2.2.1, after which it passes the request to the S-CSCF. At this point, the S-CSCF is required to lookup the iFC that determine how to handle the request. Though the iFC are part of the user profile which is stored in the HSS, the iFC should already by available to the S-CSCF since it is required to download the iFC from the HSS when the user registers. If the request is a SIP REGISTER request and an application server has been configured in the iFC to receive these requests as third party registrations, the S-CSCF forwards the request to the indicated application server. On any other request, the S-CSCF compiles a list of all the iFC for that request and orders them by their priority so that the iFC with the highest priority is handled first. It then parses the request to extract its properties and if there is a match between those properties and the trigger point in the iFC, the request is dispatched to the indicated application server recorded in the iFC. The S-CSCF repeats this process for all iFC that were compiled until the last one has been examined. Once it has exhausted all iFC, it then routes the request using normal SIP routing rules. For instance, the S-CSCF may route a response back to the UE to terminate the session. If at any point, an application server that is sent a request by the S-CSCF fails, the S-CSCF uses the value of the default handling parameter in the iFC to determine if it should terminate the call or if it should continue examining any remaining iFC that was matched. Figure 2.9 summarises the interactions that the S-CSCF has with related entities and with iFC.

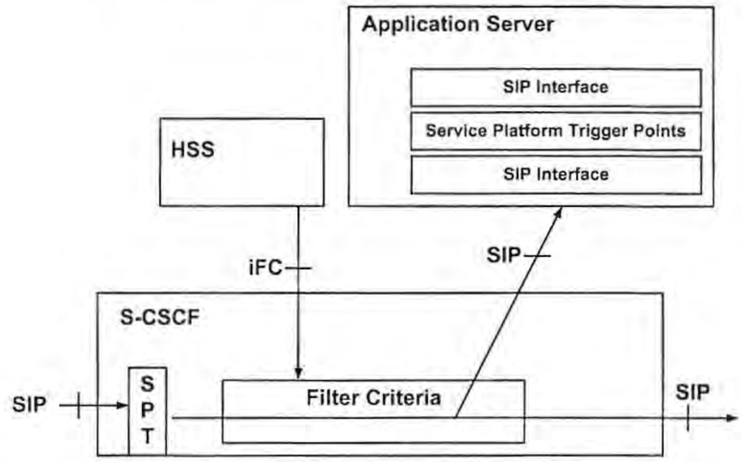


Figure 2.9: Service triggering architecture in IMS. Source: [8]

When an application server receives a request from the S-CSCF, it performs the programmed application logic that it has been provisioned with. According to [8], there are essentially four modes of operation that are exhibited by an application server. The first is as a terminating UA / redirect server where a SIP request is proxied by the S-CSCF to the application server, which then acts as either an IMS UA or a SIP redirect server. The second is as an originating UA where the application server generates SIP requests and sends them towards the S-CSCF. The third is as a SIP proxy, where the application server proxies the request back to the S-CSCF, which carries the request toward its next destination. Fourthly, an application server can behave as a third party call controller (3PCC) / Back-to-Back UA (B2BUA). A third party call controller is more sophisticated and powerful than a SIP proxy since it can receive requests and generate totally new ones, or it can create multiple SIP requests with different SIP dialogs and corelate them. For the sake of brevity, graphical representations of the different modes are provided in Appendix A.

2.3 Interaction Management and the SCIM

It is evident that there is a formal model that describes the delivery of services in IMS. This model allows for a cascaded execution of services, based on a set of priorities defined in the service profile of the user. In this model, application servers interact only with the S-CSCF, which handles all the necessary operations, including responding to error conditions. This model, however, does not seem to work very well when service capabilities are involved. Firstly, the functionality required to deliver a complete service is not contained in a single application server, but is potentially spread across several application servers in the network. In this situation, the S-CSCF must be aware of all the locations of the service capabilities that need to be invoked in order to deliver the complete service. This facility is not supported by the iFC model that is currently used since iFC only indicate the address of a single application server.

In addition, a service which is composed of different parts requires those parts to interact with each other in order to deliver the service and not only to interact with the S-CSCF. This is similar to an application server which uses conditional statements in its code to invoke certain features and disable others. Because those features are the responsibility of a single application server, it is easy to handle those cases, but in the absence of a manager that can manage these interactions, the challenge remains.

In response to this, 3GPP has defined a special function as part of the SIP application server known as a service capability interaction manager, or SCIM. This node is shown inside the SIP application server in Figure 2.6. In [8], the SCIM is described as a special server that performs the role of interaction management between multiple service units. Two forms of interaction management have been defined. One is static interaction management, which describes the use of iFC (static rules) to govern the invocation of services. The current model of service execution conforms to this type of interaction management. The other form is dynamic interaction management, which describes the use of dynamic information which cannot be conveyed using iFC to govern these interactions. There are certain cases that could benefit from the use of dynamic service interaction management, such as a service that behaves differently on a subscriber's birthday or one that takes a runtime condition into consideration when executing. These conditions cannot be handled well using the current static model, which can only use SIP-based attribute classes to inform service execution. The SCIM represents the next frontier in service execution in IMS, and promises to fully exploit the method of service creation that is based on service capabilities. The SCIM is dealt with more rigorously in the next chapter, where efforts by the 3GPP and members of the research community to define the SCIM are described.

2.4 Discussion

This chapter has covered service aspects in IMS. One of the key points that was conveyed early in this chapter was that there is a new move to describe services in terms of the building blocks, or service capabilities, that are needed to construct them. The benefits that accrue from this strategy include cost savings, time savings and the ability to provide service interworking from standardised service capabilities. This strategy is not dealt with firmly in the standard IMS service architecture, which simply recognises services as features hosted on an application server. The SCIM was introduced briefly whose function is to perform interaction management between multiple service units, though the mechanisms that support this function have not been fully developed or standardised. Provision for the deployment of multiple service capabilities is something that must be recognised and catered for by operators who wish to provide ample services to their subscribers, and to inherit the benefits that have been listed. It is clear from this chapter that the IMS technical specifications have not entirely caught up with the realities of this new shift in service creation, and this fact was highlighted in the summary on the SA WG1 mailing list discussion. In the next chapter, attempts from the 3GPP and the IMS research community as a whole to address this challenge are presented. The discussion raises important issues that result from the deployment of multiple service capabilities in the IMS, which will also be explained.

Chapter 3

Interaction Management in the IMS

In its most basic form, the IMS service architecture consists of the S-CSCF, the HSS and three types of application servers. The execution of services is supported by static interaction methods that involve the use of fixed, predefined rules that are based on the SIP protocol, and allow service requests to cascade through a set of fixed application servers on the basis of priority. This way of managing services may suffice for most cases, but experiences problems when the services that must be executed are distributed across several application servers. The 3GPP has also noticed that the current service model is not well suited for cases that depend on conditions that cannot be evaluated using the SIP protocol alone. For example, say a particular service must behave differently on a specific calendar date, such as a user's birthday or wedding anniversary. Or say a certain service must be skipped if the service that was executed before it generated a certain result at runtime. These cases require external, non-SIP conditions to be considered when executing a service and cannot be enforced using existing interaction methods. There are cases that have been documented where services behave well when executed individually, but can cause undesirable results when executed one after the other. This is known as feature interaction, and these types of interactions must be managed appropriately in telecommunication networks. The use of non-SIP conditions to influence service execution and the management of feature interactions both pose challenges to the current service model in IMS. In addition to this, there are certain opportunities that the current model simply fails to grasp. Operators may want to apply certain policies that affect the way in which services are executed in the service chain. Similarly, users may want to customise the way in which they interact with a service based on their own preferences. The application of polices would allow operators to manage feature interactions. Service personalisation would add value to the services that are offered on the network, which could have revenue-generating potential. This chapter discusses these challenges and opportunities, and shows how the 3GPP, through a technical report, tried to address them and how the research community has also contributed to this discussion.

3.1 The SCIM and the Service Broker

The previous chapter stated that a SCIM is an optional component of a SIP application server. The 3GPP included this entity in the Network Architecture technical specification with the assumption that it would be involved, somehow, in managing the interactions between service capabilities hosted on SIP application servers [46]. Given that a single physical node can host several of these, the SCIM was an attempt to provide the necessary explanation for how services that required the interaction of multiple service capabilities hosted on different physical nodes would be catered for in the IMS.

The SCIM is not the only functional entity that has been proposed for performing the role of interaction management. The 3GPP has defined a service broker as part of the OSA framework that facilitates third party access to services and network features in a managed and controlled manner. The service broker was to be a separate physical entity to the standard IMS service nodes and would include a service broker API that would allow for the configuration of data that is used in the brokering process. The 3GPP technical specification TS 23.198 does not explicitly mention where the brokering function would itself would be implemented, but indicates that it could have an interface with the OSA SCS [35].

The SCIM and the service broker have both been indicated to be involved in interaction management, but very little has been specified about their structure. To introduce the discussion on interaction management, a technical report on this topic that was published by the 3GPP will be summarised, and subsequently, the various attempts by the research community to address this topic are presented and analysed [1]. The review of proposals is divided into four sections, each of which relates to different aspects of the interaction manager. These are architectural placement, algorithms for service interaction, feature interaction management and brokering functions.

[1] The 3GPP refers in its report to the entity that performs interaction management as the service broker, and in certain cases, the research community does the same. In other instances, however, the name SCIM is used. The policy applied in this chapter is to use the term that is used in the document being described, and to default to the name interaction manager when describing this entity in a general sense. This ambiguity is clarified later in the chapter, when the functions of both entities are distinguished.

3.2 3GPP Technical Report: TR 32.810

Though the SCIM has been defined since 2002 (3GPP Release 5) and the OSA service broker since 2006 (3GPP Release 7), subsequent to the publication of the related technical specifications, the 3GPP did not invest much effort in describing the nature of interaction management where a service is composed of several service capabilities [10]. In Section 5.5.2 of TS 23.002, the only remark alluding to interaction management states that the functional architecture of the SCIM is outside of the standards. The issue of interaction management was finally visited in 2008 when a work item was initiated by SA WG1. The purpose of the work item was to investigate the potential impact of the SCIM as a component of a SIP application server, and the service broker as a component of the OSA framework, on the current architecture. The investigation was expected to shed light on the suitability of the current service interaction mechanisms through the use of iFC for managing interactions between the home network and third parties. It was also expected to determine if any changes were needed to the standard service architecture, or in the IMS protocols themselves, in order to support the two entities. The prevention of certain interactions was also listed as an item of study, as it was expected that certain interactions could be dangerous or undesirable.

The study ended with a number of suggestions being made, but at the time of writing, no further action has been taken by the SA WG1 on this topic, aside from the decision to regard the study item as complete with no new work item descriptions planned [55]. The findings of the working group are provided in the technical report TR 32.810 [56]. The major themes that arose from this report are discussed in the next few sections. Each section starts with the initial findings of the working group and is supplemented thereafter by a summary of the findings of IMS researchers on the same topic.

3.3 Architecture Alternatives

In the 3GPP technical report, the working group identifies three possible architectural placements for the service broker. It could be placed in a central position between the S-CSCF and the application servers so that the service broker interfaces with more than one application server. It could also be distributed across several physical nodes such that each service broker is assigned one application server and the individual brokers communicate with each other to manage the necessary interactions in a co-operative fashion. The working group acknowledged that the distributed placement option requires mechanisms to be put into place that ensure consistent and coherent communication between the service brokers, but none were proposed in the report.

A hybrid placement is also possible where several service brokers can be deployed as in the distributed case, but like the centralised case, a service broker can communicate with more than one application server. This definition permits two possible configurations. One is where some service brokers exhibit centralised and distributed behaviours, and another, where all service brokers are both centralised and distributed. The technical report also debates the question of the autonomy of the service broker. The view was that the service broker could either be a separate standalone entity, a component of an application server or could be merged with the S-CSCF. Regarding architectural placement, the only decision that was taken by the working group was that only the centralised placement would be considered during the investigation, and that other modes would be for future study. Appendix A provides illustrations of the different architectural placements that are described in the technical report.

Many papers on the subject of the interaction manager assume a centralised orientation in order to deal more thoroughly with structural and functional aspects, for example [57, 58, 59]. However, in their paper entitled *SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture*, Gouya *et al.* address the question of architectural placement [60]. The authors only consider the centralised and distributed models and no explicit mention of the hybrid model is made. They argue against the centralised approach, citing scalability and the *bottleneck effect* — whereby the single SCIM node becomes unable to successfully dispose of its duties in periods of high load as reasons. Due to this, the distributed model is encouraged. In their view, to ensure scalability and effectiveness, the SCIM should be distributed and incorporated into the S-CSCFs in the network, such that each S-CSCF is co-located with a single SCIM node. The authors believe that this architecture provides them with the benefit of scalability because multiple S-CSCFs can help share the load co-operatively in periods of high traffic. The authors claim that co-locating the SCIM with the S-CSCF is an efficient strategy because once the S-CSCF has downloaded the user profile for the subscribers it is responsible for the SCIM does not need to repeat this procedure using its own Diameter interface. The need for a Diameter interface at the SCIM arises from the authors' hypothesis that the SCIM would need to download user profile information for the purpose of interaction management. What is unclear with this proposal is the extent of the impact of incorporating the SCIM into the S-CSCF. However, it can be expected that this design would require an update of all IMS S-CSCFs, which is not feasible for existing networks wishing to deploy a SCIM.

In their review of this paper, Spiers and Ventura make the observation that it is possible that binding the SCIM to the S-CSCF could lead to problems should either the SCIM or the S-CSCF fail [57]. Placing the SCIM outside of the S-CSCF, while retaining the one-to-one mapping between the two, would only affect services requiring interaction management should the SCIM fail. Spiers and Ventura correctly note that this tight bundling fails to provide a seamless deployment of the SCIM, which would be in the best interest of network operators and would also minimise the impact of the SCIM on the architecture as a whole.

Qi *et al.* also make reference to [60], and while they agree that distributing the SCIM would have scalability benefits, they approach the distribution in a different manner. In their paper *D-SCIM: A Novel Service Invocation Mechanism in IMS*, the SCIM is not a component of the S-CSCF, but is a standalone system [10]. Also, it is not distributed across the S-CSCFs, but the SCIM function itself is distributed. Their SCIM is decomposed into two logical components, a centre SCIM (C-SCIM) and one or more SCIM nodes (SCIM-N). The C-SCIM is the part of the SCIM function that is responsible for interaction management, while a SCIM-N is co-located with an application server and communicates with the C-SCIM and other application servers on behalf of the C-SCIM. In turn, the C-SCIM consists of two logical functions, the Service Application Interaction Logic (SAIL) and the Service Application Register Logic (SARL). The SAIL function is core to the C-SCIM and supports operations such as the configuration of the network operator's service invocation policy, feature interaction and service composition. The internal mechanisms of DSCIM are covered in greater depth in the next section. The authors do not provide a description of the SARL, but it seems to act as a repository for service information, as the authors mention that when a new application server is deployed, its details must be provisioned in the SARL. The main motivation for this design is that the authors believed that such an arrangement would reduce the load on both the S-CSCF and the main SCIM function. The D-SCIM option would fit more naturally into the existing IMS service architecture, since the SCIM function is separate from the S-CSCF, unlike in [60]. The architecture does however delegate some of the SCIM function to the application server. The implications of this design decision are discussed in the next section.

3.4 Algorithms for Interaction Management

Interaction logic is the logic executed by the interaction manager in order to perform its duties. The 3GPP technical report introduces a possible solution that uses a set of rules that are based on two dimensional tuples of the form `<ServiceID, ServiceEffect>`. These tuples identify an application server (represented by the `ServiceID`) and an historical reference (represented by the `ServiceEffect`). The `ServiceEffect` portion conveys a runtime result that is generated after the execution of a service. Using this model, it is possible to provision the following rules which would be stored as an extension of the user profile in the HSS as service-related information to be used for interaction management:

- If History contains <Service A, Success>, SKIP Service B
- If History contains <Service B, Failure>, SKIP Service C
- If History contains <Service B, Failure>, SKIP Service D

The working group recognised the importance of formulating a model that would work in concert with the existing service infrastructure, particularly with regard to the re-use of iFC. Retaining the iFC would mean that this portion of the current model would not be made redundant. The service broker would record services and effects to provide a model that uses historical data to determine subsequent targets in the execution chain so that service execution is influenced by non-SIP conditions and runtime values.

This approach introduces two complementary forms of interaction management, online and offline. Offline interaction management refers to the mechanisms that are needed to manage historical rules, such as those listed above. Since this occurs outside the context of an actual communication session, these mechanisms fall into the offline category. Online interaction management refers to those mechanisms that are involved in handling a live session, where those historical rules are taken into account. Both offline and online methods are necessary to fully perform interaction management.

In another paper by Gouya entitled *Managing Service Capability and Service Feature Interactions in the IMS of UMTS*, a solution for interaction management is presented [52]. In this model, when the S-CSCF receives a SIP request, it examines the iFC that apply. If the iFC indicate that the request is for the invocation of an integrated service, it adds a SIP dialog identifier to it and proxies it to the SCIM. When the SCIM receives the request, it proxies it to the indicated application server. When the application server is done processing the request, it returns it to the SCIM. The SCIM attaches a service capability profile in the body of the SIP request and forwards it to the S-CSCF. The service capability profile, or SCP, is similar in structure to the IMS service profile, except that it maps an application server to zero or more service capabilities in order to communicate service composition information. The SCP is stored in the HSS and is retrieved through an Sh Diameter interface. The S-CSCF uses the information from the user's SCP to learn the next hop based on the service composition and sends it to the indicated destination. The motivation for the use of the SCP is that it provides a common and flexible template that can be used by all service providers to describe their service compositions. The call flow for the interaction is depicted in Figure 3.1.

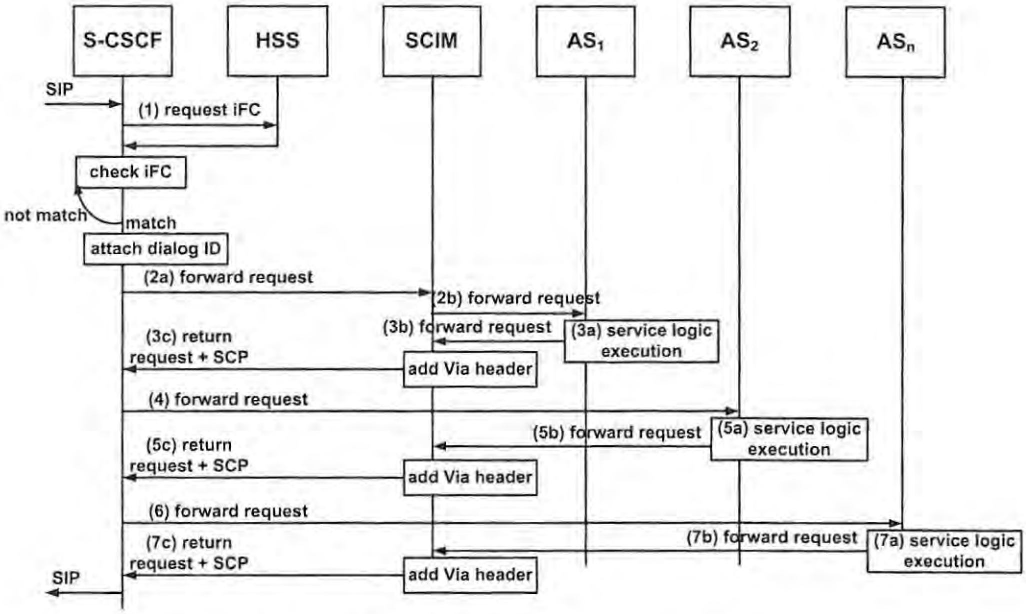


Figure 3.1: Service interaction model 1.

The DSCIM system offers an alternative solution. In this case, when an application server has finished performing application logic based on a SIP, the co-located SCIM-N component copies the request and proxies it to the CSCIM. The CSCIM in return will modify the Request-URI based on iFC and send the request back to the SCIM-N. The SCIM-N forwards the request based on the indicated address in the Request-URI header to the next SCIM-N and so on. This process is repeated until all matched iFC are evaluated. When all criteria have been evaluated, the request is forwarded back to the S-CSCF to indicate that the execution of the service has been completed. The main difference between this model and the one previously described is that in DSCIM, the S-CSCF is no longer an active participant in the call flow. Figure 3.2 provides a graphical representation of how the CSCIM and SCIM-N work together with the S-CSCF.

The paper by Wang *et al.* entitled *An Improved SCIM-based Service Invocation Mechanism for Integrated Services in IMS* describes another interaction model [53]. Here, the authors consider an integrated service to be one that consists of a core service capability and one or more auxil-

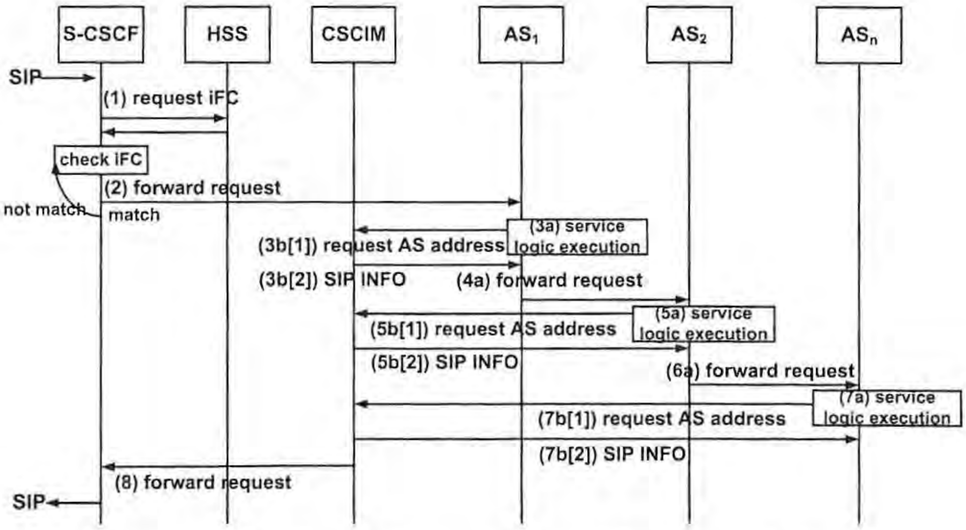


Figure 3.2: Service interaction model 2. Source: [10]

iary service capabilities. The core service capability performs most of the functionality and the auxiliary ones perform the less important tasks. Their proposal also addresses offline interaction management by providing a facility for iFC to be uploaded to the SCIM by the S-CSCF when a user registers. During online interaction management, the request is sent first to the SCIM as an intermediary between the S-CSCF and the services, unlike in DSCIM where the initial request is sent directly to the application server itself. The SCIM examines the iFC and forwards the SIP request to the core service capability which applies service logic and returns the message to the SCIM. Unlike with DSCIM, the service capabilities do not assist the SCIM in routing requests to each other, but return the request to the SCIM each time until all iFC are evaluated. Lastly, the request is sent to the core service capability and then to the SCIM and finally to the S-CSCF. Figure 3.3 shows the corresponding call flow for this scenario.

These three proposals share some similarities but are certainly different. From them, it is possible to extract four main themes. Firstly, offline interaction management can be used to assist the SCIM in obtaining the rules that it needs to invoke multiple service capabilities in preparation for the online stage of service execution. This is pertinent in cases where the SCIM, as opposed to the S-CSCF, is the principal routing agent for online interaction management (which is the case in the last two proposals). Secondly, when an integrated service is executed, the initial SIP request can either be sent to the SCIM or directly to the service capability with the highest iFC priority. The proposal related in [10] supports the latter, whereas [53, 52] support the former. Thirdly, once a service capability has executed its task based on a submitted request, it can do

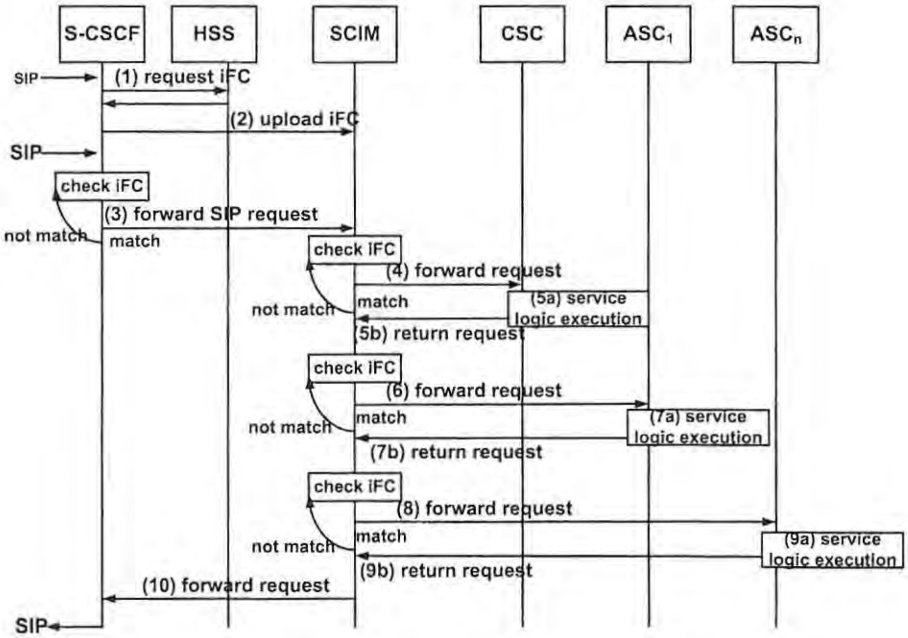


Figure 3.3: Service interaction model 3.

one of two things. Either it can participate actively in the forwarding of the request to the other service capabilities or it can delegate that job to the SCIM or to the S-CSCF. Interestingly, each of the three proposals takes a different stance on this issue. Finally, data besides the iFC alone (such as the SCP) can be used to provide additional information for handling interaction in the case of integrated services. However, while the use of data such as SCP can be beneficial, the designer must think carefully about the balance between iFC and these forms of information. The solution must address how they interact with each other and which one takes precedence over the other. For example, in [52], the iFC are used to determine the forwarding of the initial request from the S-CSCF to the SCIM, but subsequent to this, only the SCP are used for routing.

In [53], some insight on the issue of offline service interaction management is provided through the proposal that in certain cases, such as upon user registration, iFC should be uploaded onto the SCIM. This is intuitive and is in fact already done by the S-CSCF [8]. However, the authors in [53] do not detail how this should be done. A solution could be to allow the SCIM to download the user profile and to register for user profile updates from the HSS so that when a user registers or deregisters, the correspondent iFC are uploaded onto the SCIM. The forwarding of the initial request for an integrated service is probably best addressed by sending it to the SCIM, as is the case in both [10] and [53]. This makes sense from a SIP transaction point of view, as when the SIP request is forwarded to the SCIM from the service capability, the SCIM would not have created a dialog with this server beforehand. It would also allow the SCIM to perform additional logic that arises from dynamic conditions that cannot be reflected by the static iFC rules which the S-CSCF would be basing its actions on. The routing of requests through the chain of service capabilities is also probably best handled in the manner outlined in [10] since the load on the S-CSCF and the SCIM is significantly reduced through the co-operative sharing of responsibilities by the service capabilities themselves. In fact, it is unclear why the S-CSCF should be involved in the call flow as it has no obvious benefit. By inserting another node into the call flow, the total number of message exchanges is increased, which in turn increases the load on the network. Both the S-CSCF and the SCIM should have access to the SCP through the Sh interface, thus the forwarding of the SIP request with the attached SCP to the S-CSCF seems redundant. By observing the number of message exchanges in the two diagrams, it is evident that the approach advocated in [10] is less burdensome on both the S-CSCF and the SCIM.

The introduction of the SCP is important as it raises the issue of how limited the current service profile in IMS is as it is not well-suited for showing how a single service can be composed of multiple service capabilities residing on different application servers. The difference between relaying a request through a static chain of application servers and relaying a request dynamically through a chain of service capabilities is subtle, but important. It would seem, and this sentiment is reflected in the DSCIM system, that a request should be relayed to the service capabilities in sequence before being sent back to the S-CSCF for evaluation against iFC. This follows since the signaling between the service capabilities constitutes the execution of a single service. When the last service capability is invoked then the composite service is satisfied and normal routing of the request can continue using default SIP rules.

3.5 Service Interactions

3GPP's technical report recognises several interaction types such as interactions between applications, interactions between users and interactions between service provider domains. In these scenarios, there can be desirable and undesirable interactions. Undesirable interactions are those that can cause undesirable results, whereas desirable interactions are those that produce results that are expected by the user or developer. Undesired results can arise due to unknown conditions that may only manifest when applications execute one after the other. The execution of one service may also inhibit the execution of another. Undesirable interactions such as these must be avoided at all costs.

An example can be made with a call service that provides subscribers with originating call screening (OCS) and call forwarding (CF) features. If the user Alice wants to call another user Bob, then that call should be terminated as expected, unless if Bob has activated the CF feature on his profile. Assuming that Bob forwards his calls to Mallory, Alice will be forwarded to Mallory when she attempts to place the call. There need not be anything wrong with this case, except if Alice has activated the OCS feature on her profile in such a way that all calls to Mallory are screened. The case of Alice placing a call to a screened user thus causes an undesirable interaction involving the OCS and CF features. An interaction manager would be expected to support the execution of services in such a way that it prevents undesirable interactions such as these from occurring. The challenge of application interaction is not limited to the IMS environment, but has been a challenge in legacy telecommunication networks and more recently in new Internet services as well [61]. Application interaction is commonly referred to as feature interaction and is sometimes abbreviated as FI.

Li and Yang make note of this point in their paper *A Generic Approach to Service Conflict Control in IMS* and suggest that the IMS subsystem, due to its emphasis on convergence and its utilisation of numerous IP protocols, presents a greater challenge than legacy networks [62]. In legacy networks, the management of feature interactions is normally restricted to call scenarios while in IMS, many different types of services can be involved besides voice or video calls, and many protocols can also become entangled in a conflict. Also, since IMS service provisioning can be informed through user personalisation through XCAP policies, IMS exposes itself to user-orchestrated conflicts as well. Li and Yang's solution architecture, which they name GICC (Generic IMS service Conflict Control), is modular in nature and consists of five distinct components. A service behaviour description module is utilised that works in concert with an information abstraction module to transform an incoming message into a technology-independent format. The service description data itself is expressed in an object-oriented manner, and comprises of `Event`, `Operation` and `Parameter` parts. An `Event` can be an incoming call or a timer expiration, an `Operation` is any action that can be performed on the event such as forwarding the incoming call, and the `Parameter` stores environmental values such as time of day or location. Another module called the interaction modeling module receives the formatted message and generates runtime service behaviours, while the main GICC module uses constraint rules to evaluate and enforce conflict control. The authors provide a mathematical proof based on predicate calculus that shows that theoretically, their solution could be used to detect conflicts before they occur and approve resolutions to avoid undesirable outcomes.

The GICC framework provides several important insights into application interaction. For instance, the service description feature provides an alternative to iFC. This approach is richer than the iFC model and allows a very detailed way for operators and service providers to describe services, while resolving conflicts in them at the same time. The generic message format also provides a convenient way of addressing the convergence problem since it defines a common format for programming conflict resolution in a multi-protocol IMS. The constraint ruleset would be an important aid to the SCIM in evaluating and enforcing application-level policies.

In his paper, *A Distributed Mechanism to Resolve Dynamically Feature Interaction in the UMTS IP Multimedia Subsystem*, Crespi re-iterates some of the sentiments expressed by the creators of the GICC system [63]. Here, the author states that due to the open nature of IMS through service interfaces, feature interaction management will be more complex since new application servers are constantly added by operators and third parties. Crespi also recognises the importance of offline interaction management but notes that some undesirable interactions cannot be mitigated by offline techniques. These specific types of interactions are divided into two types, those occurring within one domain (handled by one S-CSCF) and those across multiple domains.

Using examples of services, Crespi shows how undesirable interactions can occur with certain services such as with a terminating call screening (TCS) service and a call forward on busy (CFB) service. It is possible in this case for a call that would have been screened by the TCS service to be allowed to proceed if the intended callee has activated the CFB service which would forward the call to the screened user. To cope with such scenarios, Crespi proposes that services be provided with a description of which modifications to the SIP messages should not be permitted for the next hop, through the introduction of a new SIP header called `ServiceRule`. The rule has three parts, which are `Applicability, MessagePart` and `ForbiddenValues`.The `Applicability` part specifies the requests or responses for which the rule applies. The `MessagePart` specifies the elements affected by the rule, such as a SIP header or SIP message content. The `ForbiddenValues` part consists of a pattern matchable string that specifies the values that the elements in the `MessagePart` must not be set to. An example is:

$$ServiceRule: \mathit{Applicability} = 480;\ MessagePart = RequestURI;\ ForbiddenValues = all$$

The above rule indicates that if the application server responds with a SIP 480 response code (Temporarily Not Available) the SCIM is forbidden to forward the request to another destination by changing the Request-URI header.

Usually, extensions to the SIP protocol to handle niche applications like this are discouraged. The relationship between 3GPP and the IETF has been highly formalised in order to control both the number and the nature of the modifications to SIP (see Section 1.2.2.1). Fortunately, this is not the case here since the `ServiceRule` header is not acted upon by the IMS Core entities, but by the SCIM only. An interesting contribution that Crespi has provided is that the service descriptions he specifies contain forbidden behaviours. This does pose a challenge since it is inconceivable that a developer would not likely be able to know in advance all the different cases that would lead to undesirable behaviours. But since the rules are essentially extensible, it provides room for incremental consolidation of such rules after conflicts have been detected and become known.

3.6 Brokering Functions

The 3GPP consistently uses the term service broker in their technical report to refer to the entity that is responsible for the functions that have been attributed in the Network Architecture technical specification as the responsibility of the SCIM. As the technical document that was primarily responsible for setting the tone in this area, this has led to confusion in the research community. It is possible, however, to differentiate between the two and allocate specific functions and responsibilities to each.

The Fokus Fraunhofer Institute is one of the major research institutions in Germany that conducts work in the area of IMS [64]. As part of their efforts, they have also published a number of articles on the topic of interaction management. They take the view that the entity known as the SCIM is only a part of the solution to the overall problem, and can in fact be regarded as a component of the service broker.

In a paper authored by researchers at Fokus Fraunhofer entitled *A Service Broker Providing Real-Time Telecommunications Services for 3rd Party Services*, Blum *et al.* present the results of a collaborative effort between Fokus Fraunhofer and the German operator Deutsche Telekom [65]. In it, the authors detail the relationship between modern mobile IP networks and the Internet. The authors assert that the challenge of interaction management in IMS is not limited to services in the home domain alone, but since the network has interfaces that allow external access, the involvement of third parties also contributes to the problem. In this climate, catering for operator-defined policies and securing the service capabilities that are exposed, are both listed as important objectives within the wider framework of interaction management. In their paper, the authors provide a high level design of their service broker, and list the following as key components:

Policy evaluator The main service broker component which identifies policies that apply to an incoming message

Policy enforcer The component that executes the decisions made by the policy evaluator

Policy manager The component that facilitates the management (upload, modification, deletion) of the policy repository based on the XCAP protocol

Workflow engine (service delegation) An engine which is based on the Business Process Execution Language (BPEL) that supports operator-defined policies that affect the processing of a request towards a third party application server

Service registry The repository that stores details of known services

Service capability interaction manager The component that allows the dynamic adaptation of services based on user, service and network domain identifiers

The service broker presented by Blum *et al.* is inspired by work that was conducted by the OMA. The OMA found itself in need of a unifying architecture to complement their collection of over 100 service enablers. The uncoordinated deployment of such enablers would be overly complicated and would inevitably lead to high implementation costs for services seeking to use multiple enablers. Thus the OMA defined the OMA Service Environment (OSE) which consists of elements including enablers, and defines the relationships and interfaces that exist between those elements. The architecture of the OSE is given in Figure 3.4.

Figure 3.4 shows that applications can reside both inside or outside the OSE domain, much like the IMS. The policy enforcer in the OSE combines the functions attributed to both the policy enforcer and the policy manager in [65]. It does so by applying the policies that govern the appropriate use of enablers and managing the requests that are delivered to it. The policy enforcer also provides support for billing, generates logs and enforces user preferences and privacy settings. It is loosely based on an existing OMA service enabler known as the Policy Evaluation and Enforcement Manager (PEEM) [66], though the OMA does not mandate for the PEEM to be used as the policy enforcer. The OSE does not make any restrictions on how the enablers are implemented, but requires them to provide standard functions that can be exposed using some defined method. The bindings that are shown in Figure 3.4 are platform-specific ways of providing access to the enabler's exposed methods. Thus web services, Java, C and others can be developed for this purpose. The execution environment provides software lifecycle management

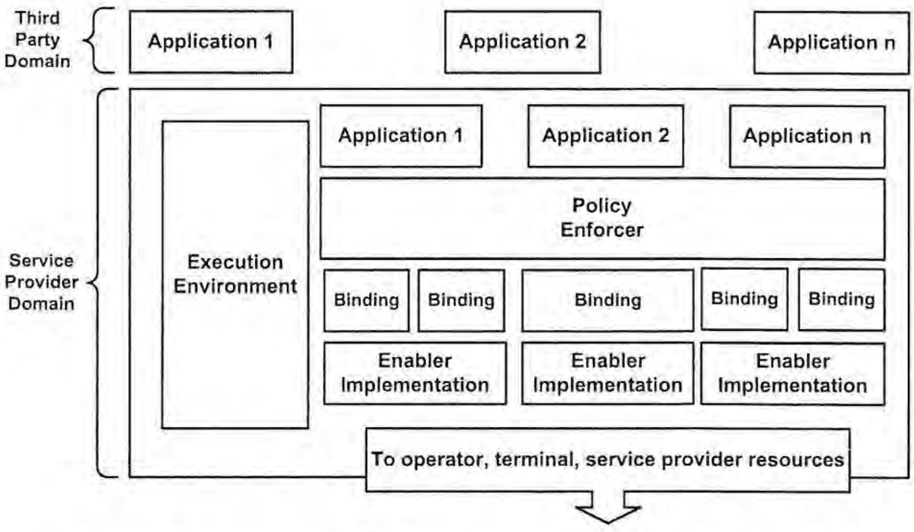


Figure 3.4: The OMA OSE architecture. Adapted from: [11]

interfaces that allow an operator to create, deploy, remove, upgrade and perform other functions related to service enablers.

The solution that is proposed in [65] is a very comprehensive one that shows that interaction algorithms, such as those that have previously been described, are only a part of the overall solution towards interaction management. There are similarities, however, between their service broker and the ones that have previously been described. For example, GICC has constraint rules and conflict control modules which are similar in principle to the policy evaluation and policy enforcement modules. GICC also has a service description module which is similar to the service broker's service registry. This service broker differs slightly with regards to policy issues in that it describes business level policies, permissions related to the access of service capabilities and user preferences instead of just the policies that determine the basic rules of service interaction. In contrast, the GICC system approaches policies from the perspective of service conflict avoidance and not the protection of service capabilities, though these two policy types are complementary. It is thus conceivable that the modules introduced by the service broker can be extended to support and cater for service conflict avoidance and not only for securing the service capabilities from misuse and honouring user preferences. This would give a fuller definition to policy evaluation and enforcement.

In a follow up to [63], Gouya and Crespi presented a paper entitled *Service Broker for Managing Feature Interactions in IP Multimedia Subsystem* which combines the realisation of a service broker with feature interaction [12]. The proposed architecture is shown in Figure 3.5. The solution

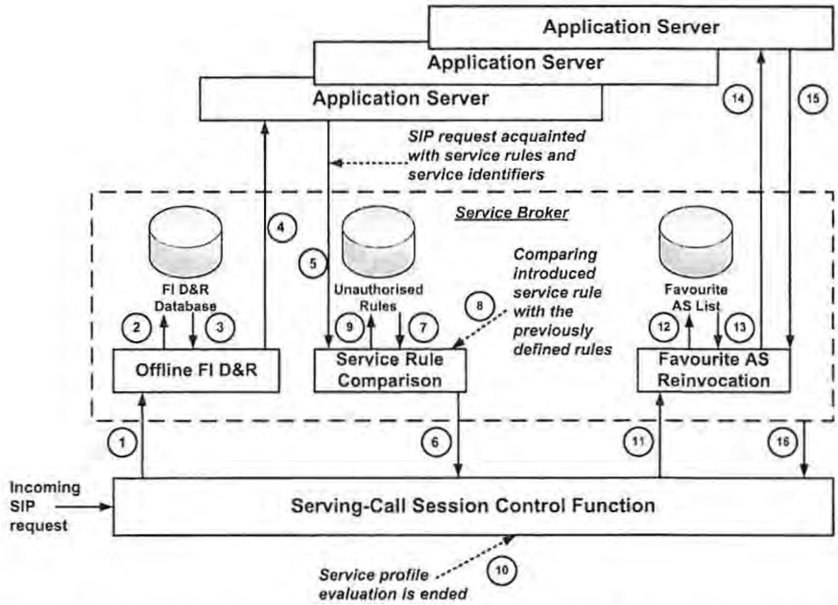


Figure 3.5: Feature interaction and detection using a Service Broker. Adapted from: [12]

is centred around feature interaction detection and resolution, network policies and user preferences. The FI D&R (feature interaction detection and resolution) repository maintains a record of services and the known conflicts they have. The unauthorised rules repository stores operator-defined policies regarding the protection of service capabilities from mis-use by home network and third party application servers. If a requested action is in violation of the network policy, that message is dropped. The favourite AS list is a user-defined list that records the application servers for which the response from the last application server invoked must match the response from the indicated application server in the favourites list. A use case for this is illustrated in the paper where the authors give an example of a user who is barred from international calls. Assuming that the user is a subscriber of an operator-assisted service for making international calls, a conflict that could arise from allowing a call to a barred location through a third party could be avoided by comparing the SIP dialog states of the SIP message that is to be sent out by the operator (which contains an international number) and the actual one being made (which contains a local number). Consequently, the call barring feature would not permit this operation and the request would be rejected. This service broker is dissimilar to the one described by the Fokus researchers in that it focuses mostly on feature interaction detection and resolution. The provision of a repository for operator policies would be an improvement to Gouya and Crespi's proposal which defines policies only in light of FI D&R.

3.7 Discussion

This chapter has covered three practical problems that result from the composition of services from their basic building blocks through a SCIM. It is intuitive that distributing the SCIM increases its availability, however this decision must be weighed against the complexity that multiple SCIM nodes introduce into the network. For experimental purposes, and to illustrate core concepts, a single central SCIM has proven to be sufficient in much of the literature that has been reviewed. Both offline and online interaction mechanisms have been discussed. The benefit that offline interaction management provides has been well motivated and must be regarded as an important part of the overall solution. Offline interaction management calls for the establishment of data repositories in order to store information needed for the online stage. Online interaction management should utilise existing infrastructure such as iFC and incorporate repository data in order to affect the invocation of services with dynamic rules and policies that modify the default behaviours. Feature interaction is another challenge that must be addressed, and the extension of data repositories can assist in capturing information needed to cater for scenarios that must be avoided. These repositories must be extensible so as to introduce new rules that could not have initially been anticipated. The ambiguity concerning the terms SCIM and service broker was clarified by showing that researchers have identified additional functions and requirements to those that are provided by the SCIM, which fall within the domain of the service broker. Finally, it was deduced that the SCIM is a component of the service broker, whereas the service broker is concerned mostly with handling service requests from outside the network and correctly applies policies and security checks to ensure the safe exposure and use of service capabilities. The next chapter capitalises on the material that has been reviewed in both this chapter and in chapter 2 in order to formulate a novel design for the IMS service layer that supports service capabilities, and uses a SCIM for service interaction management. The design incorporates views on interaction management that this thesis holds to be beneficial, and includes some novel additions that have either not been raised or are only marginally explored in current literature.

Chapter 4

Introducing the Extended IMS Service Layer

The current IMS service model is in need of interventions and extensions in order to equip it with the functions and supporting infrastructure it needs to carry out personalisable service capability interaction management. In order for this to happen, a configurable SCIM must be deployed which is able to provide both online and offline forms of interaction management. The SCIM must be able to alter the execution of a service using both runtime conditions and user preferences. When such an entity is deployed, it will inevitably have an impact on the rest of the service layer, which implies that the architecture of the service layer will change. This chapter introduces a novel design of an extended IMS service layer (EISL) that embeds a SCIM. The purpose of this chapter is to express the design decisions that have been made in the areas of service decomposition, feature interaction management, service interaction management and service personalisation, and to portray them in the form of a service layer design. The design is inspired by the 3GPP technical report and proposals from the research community, which have been presented, analysed and contrasted in the previous chapter. This process has led to the definition of features, mechanisms and protocol interfaces which must be incorporated into the SCIM. Each extension and introduction of a new entity is described, and explanations are given that detail how existing network nodes interact with the SCIM.

4.1 Design Requirements

In formulating a design for an EISL, there are certain considerations that have to be made. Firstly, matters regarding the extension of the standard IMS service layer are the responsibility of the relevant bodies involved in the development of technical specifications for IMS. Thus the design must align itself as closely as possible to currently existing standards. This is not to say that proposals that deviate substantially from the technical specifications are of little or no merit in this discussion, but by aligning itself with established concepts, the design is automatically leveraged by a large body of work developed over a long period of time, and has also gained acceptance in the market. In addition, introducing minimal impact on the standard service layer will help guarantee that existing networks will interoperate successfully with the proposed changes.

It is also important to acknowledge the work that was initiated by the 3GPP and summarised in the technical report on interaction management. The design presented in this thesis is required to be developed in such a way that it follows from the ideas conveyed in the technical report as the 3GPP's initial effort towards describing SCIM functions. However, as the technical report is incomplete and doesn't answer all the questions, provision for proposals that are not explicitly mentioned by the 3GPP is necessary. That said, it is important that the design re-use standard IMS protocols and mechanisms and where existing IMS nodes are concerned, to introduce as few modifications as possible, if any, to them. This would act to better justify the introduction of the SCIM as a node with minimal impact on the overall architecture.

4.2 Solution Architecture

The solution architecture for EISL is depicted in Figure 4.1. The broken lines represent standard IMS interfaces, while the full lines represent interfaces that are necessitated by the design. It is intended that implementations of the design will re-use standard IMS protocols to implement these new interfaces. During the course of this chapter, it will be shown how this objective could be realised in practice.

The existence of a network administrator is assumed who is tasked with certain responsibilities by the network operator. None of these duties are explicitly required as a result of the extended design, and should be catered for by the operations support system (OSS) that is already in place in the network domain. The administrator is required to provision users into the HSS and to create service profiles for them. This is standard. New application servers such as the SCIM are

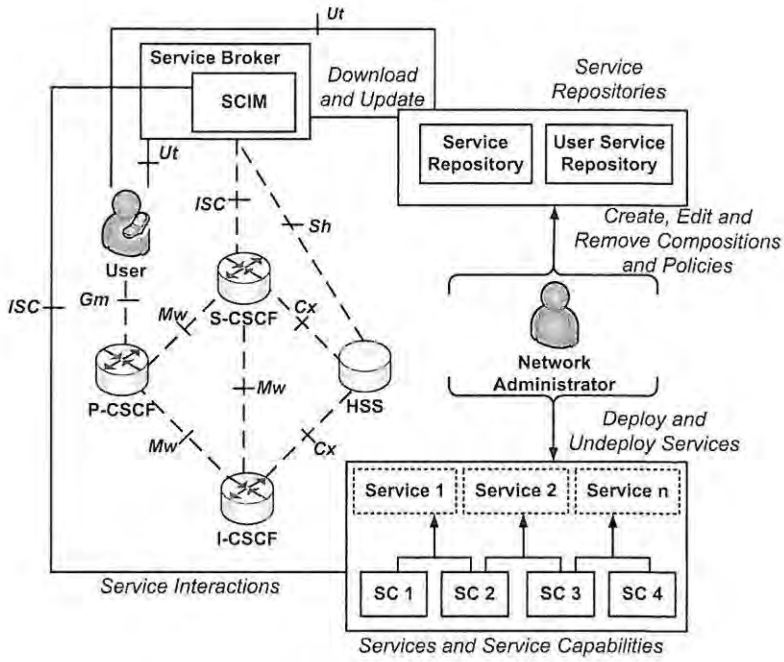


Figure 4.1: Solution architecture for EISL.

subject to deployment, configuration, monitoring and removal, much like all other services in the home network. This also applies to service capabilities that are hosted on the application servers themselves. The administrator must provision details regarding the composition of services into repositories which are similar to databases, and there will be databases in an operator's network.

The SCIM is represented as a component of the service broker. The pluggability of the service broker is ensured by the fact that entities maintain their standard interfaces. The SCIM appears as an application server to the IMS Core, and the SCIM mediates between the IMS Core and service capabilities. The SCIM is also involved in facilitating the personalisation of user services. An alternative which involves the user managing this interaction without the involvement of the SCIM is also discussed later in this chapter. The SCIM must also use an interface to download and update service information in the service repositories. The customer in the design is in possession of an unmodified IMS UE with standard protocol interfaces.

4.2.1 Service Repositories

The service repositories contain information about the communication services that are available on the network. Part of this information is provided by an administrator who may use out of band mechanisms to populate the repositories with records as show in Figure 4.1. This section describes the technologies that could be used for the repositories and provides more information about the types of repositories that are defined.

4.2.1.1 Underlying Technology

To provide flexibility of choice for implementers, the design does not mandate a specific technology be used for the development of the repositories. However the choice of implementation, in keeping with the design requirements that have been stated, should ideally be based on existing IMS protocols. In order to simplify the design, the same technology should be used for the different repositories, although this need not be the case.

The purpose of the system repositories is to store service-related information. Obviously, the need to provide data storage and management is not a unique requirement that is introduced by this design. The HSS, for example, provides a similar function by storing user profiles. The HSS is a master database that is responsible for storing comprehensive information about users and data that aids in call and session management. It would be tempting to merge the new information which is needed for interaction management into the existing HSS. A possible place to store this information might be in the user profiles of IMS subscribers. However, this approach is discouraged on the basis that it would violate the design goals previously mentioned regarding the overall impact of introducing the SCIM. An alternative solution which is encouraged by the design is to allow the SCIM to work in concert with the HSS and to supplement existing information with new interaction information.

An alternative to the HSS is the XDMS. The use of XML files as a storage medium fits the data storage requirements, and furthermore, XCAP also provides the necessary data manipulation features for adding, updating and removing records. In addition, XCAP was designed to be a lightweight protocol, unlike the heavy Diameter protocol.

4.2.1.2 Types of Repositories

In EISL, there are two distinct repositories:

Service Repository The service repository stores information related to the composition of services from service capabilities. A unique identifier is used to identify a service, each of which can be associated with one or more service capabilities. Each service capability belonging to a service is assigned a service execution priority that is used during online interaction management. Operator policies are also assigned to the services in order to implement best practices and prevent undesirable feature interactions.

User Service Repository The user service repository supports service personalisation and informs the SCIM on how to execute services in a way that is desired by the user. This is in contrast to the service repository, which merely contains service composition information and has no direct relationship with actual users. Initially, each user automatically inherits default compositions from the service repository. Should a user wish to modify the properties of a service subscription, a change in this information must be effected. Thus, the user service repository contains editable, user-defined, service composition information for a set of subscribed services.

4.2.2 The SCIM

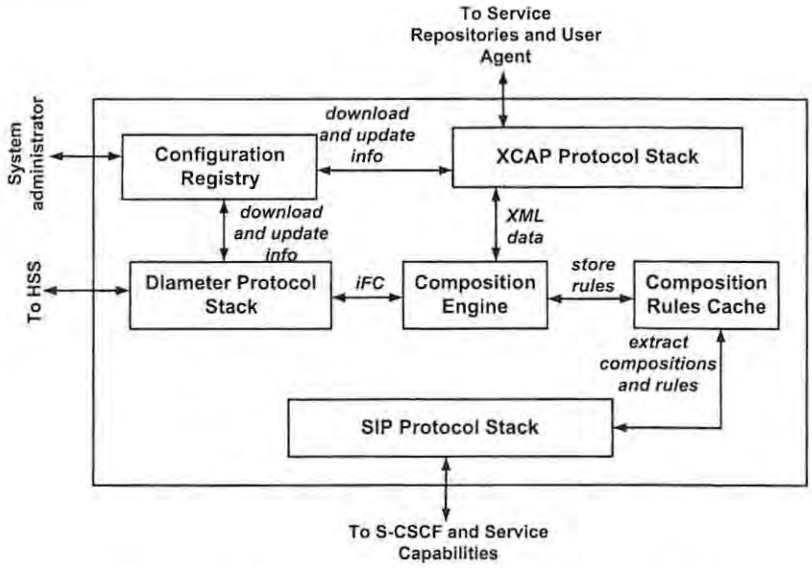


Figure 4.2: Structural architecture of the SCIM.

Figure 4.2 shows the proposed structure of the SCIM. It shows that there are three protocol stacks in use. These are SIP, XCAP and Diameter. The SIP protocol stack allows the SCIM to interact with the S-CSCF and the service capabilities through the ISC interface. Section 3.4 discussed at length the special role that the SCIM should play in the interaction signaling path and this interface allows the SCIM to do this. The Diameter stack is used for the interactions that the

SCIM has with the HSS by providing the Sh interface. Diameter requests are made using this interface, and all relevant and available HSS data is returned to the SCIM along this interface as well. The XCAP stack allows the SCIM to interact with the XDMS that provides the system repositories. The SCIM requires a local storage space which acts as a temporary storage space that stores the composition rules that must be applied for a user for which the SCIM is executing a service. The configuration registry is provisioned with configuration information by the network administrator using an out of band mechanism. This registry does not contain any service-related information, but may be used to provision other details of importance. The Configuration Repository should contain network location information related to the HSS, service repositories and the service capabilities. Other items of information may also be added at the discretion of the network operator.

The structure of the EISL SCIM differs from the DSCIM system that was presented in Section 3.3. The latter suggests that the SCIM should have a centralised controller (C-SCIM) and one or more auxiliary functions (SCIM-N nodes) that are hosted on application servers. The design proposed in EISL better fits the design requirements since the SCIM becomes a pluggable node in the service layer, and contains within itself, all the functions that it needs to perform interaction management. By relying on application servers that must be equipped with additional application logic to perform specialised routing on behalf of the SCIM, the solution becomes more involved and costly to implement. The SARL and SAIL components of DSCIM, however, are compatible with the Composition Rules Cache and service repositories of EISL. The SARL and SAIL host service composition logic and service information respectively, and as such have equivalents in the EISL system. That said, EISL's SCIM does not host service information by itself, but obtains it from the service repositories.

4.2.3 Feature Interactions

The prevention of undesirable interactions between service capabilities is an important function. Central to the management of feature interactions is the ability to obtain, enforce and amend the ordering of the interactions for a composite service. Information on the composition of services is specified in the service repositories, thus the records in those repositories must reflect the most appropriate service invocation ordering. In particular, the service repository is the correct location to implement policies for feature interaction because it is a global repository. By inserting policies there, they can automatically be inherited in the user service repository which is just a copy of the global repository with user preferences factored in. Since these rules prevent undesirable interactions in a general sense, and not from the perspective of the user, these policies must always be copied into the user service repository and must not be editable by the user.

In addition to this, the SCIM must maintain an historical vector that records the series of service invocations that have occurred during a live session. This historical vector can be used to support the management of feature interactions by comparing the service that is to be executed in the next step with the service that was executed in the previous one. Illegal interactions can thus be caught by the system and avoided.

4.2.4 Interaction Management

It is worth re-iterating that the SCIM does not need to be used for all service requests, but only for those that specifically require interaction management. This design choice reduces the load on the SCIM so that it only needs to handle service requests for composed services. For all other requests, static interaction management rules still apply. When a service is handled by the SCIM, the iFC must be configured to point to the SCIM (an action performed by the administrator) as the designated application server that must handle that request.

During the online interaction stage, the SCIM extracts the user identity contained in the SIP request and compares it with stored data in the Composition Rules Cache to see if it already has access to the composition rules for that user. The stored data is usually generated during the offline interaction stage, but can also be generated automatically during the online stage if the data does not yet exist. The composition rules can be generated during the online stage just before a new request is serviced, but once it has, those rules can be stored in the cache in order to be consulted in the future. In this way, the automatic creation of composition rules is an operation that can be performed either during the online or offline stages, depending if composition rules exist for the served user.

4.2.4.1 Offline Interaction Management

Offline interaction management occurs outside the context of a live session and its objective is to create composition rules that will determine the handling of a user request at a later stage. Section 3.4 discussed strategies for the derivation of composition rules from the iFC contained in the user profile. The discussion indicated that the SCIM must be able to download this information directly from the HSS using the Sh interface. During the offline stage, the occasions that would warrant the SCIM to download the iFC are if a new user registration has been detected or there has been a change in the iFC of a served user whose requests are currently being handled by the SCIM. These conditions would result in the creation of new rules or the modification of existing rules for the user's services.

Since the Sh interface is the standard way for an application server to interact with the HSS, it is necessary to examine the properties of the Sh interface in order to choose capabilities that can be used to perform this type of operation. Table 4.1, provides a list of the command codes supported by the Sh interface. They consist of four request commands and four corresponding response commands. The User Data Request (UDR) command is sent from the client to the server in order to request user data. The Profile Update Request (PUR) command is sent from the client to the server in order to update user data on the server. A Subscription Notification Request (SNR) command is sent from the client to the server in order to request notifications of changes in the user data, and the Push Notifications Request (PNR) command is sent from the server to the client in order to notify the client of changes in user data. The information that is requested is appended to the response message in the form of one or more attribute value pairs (AVPs). The Sh interface defines several AVPs that are used for this purpose, some of which are listed in Table 4.2.

Command Name	Abbreviation	Command Code
User-Data-Request	UDR	306
User-Data-Answer	UDA	306
Profile-Update-Request	PUR	307
Profile-Update-Answer	PUA	307
Subscribe-Notifications-Request	SNR	308
Subscribe-Notifications-Answer	SNA	308
Push-Notifications-Request	PNR	309
Push-Notifications-Answer	PNA	309

Table 4.1: Diameter commands defined by the Sh interface. Source: [9].

Attribute Name	**AVP Code**	**Value Type**	**Description**
User-Identity	700	Grouped	Contains either a Public Identity AVP or MSISDN AVP
MSISDN	701	OctetString	Contains an MSISDN (Mobile Subscriber Integrated Services Digital Network) number in international number format
User-Data	702	OctetString	Contains the user data requested in the UDR/UDA, SNR/SNA and PNR/PNA operations and the data to be modified in the PUR/PUA operation
Data-Reference	703	Enumerated	Indicates the type of requested user data in the UDR and SNR operations; legal values include IMSUserState(10), iFC(13), S-CSCFName(12), and MSISDN(17)
Subs-Req-Type	705	Enumerated	Indicates the type of the subscription to notifications request; legal values are Subscribe(0) and Unsubscribe(1)
Identity-Set	708	Enumerated	Indicates the requested set of IMS Public Identities; legal values are All-Identities(0), Registered-Identities(1), Implicit-Identities(2) and Alias-Identities(3)
Expiry-Time	709	Time	Contains the expiry time of the subscriptions to notifications in the HSS
Send-Data-Indication	710	Enumerated	If present, indicates that the sender requests the user data; legal values are User-Data-Not-Requested(0) and User-Data-Requested(1)
Server-Name	602	UTF8String	Contains a SIP URL used to identify an application server
Public-Identity	601	UTF8String	Contains a Public User Identity

Table 4.2: Subset of Diameter AVPs for the Sh interface. Adapted from: [15].

Of the Diameter commands that have been described, the UDR and SNR commands best provide the functionality that is required to support offline interaction management. The UDR command is used to request user data, while the SNR command is used to request notification of changes to that data. The UDR request can be prompted by the registration of a served user, for example, in order to instruct the SCIM to compile the composition rules for that user. The iFC in the HSS could be used to achieve this, where a third party registration is created for the SCIM, such that upon receipt of a SIP REGISTER request, the S-CSCF sends it to the SCIM. The SNR command can be prompted by the successful download of user data through the UDR request, whereby the SNR guarantees the delivery of subsequent changes to that data in future. Changes in the user data may lead to a new service being allocated to the user, or a service being removed from the set of subscribed user services, which is what makes this command particularly important. The UDR and SNR commands would be used in conjunction with the `Data-Reference` AVP, with an enumerated field value of 13 to indicate iFC. In addition to this, other AVPs that would be included in these requests are `Subs-Request-Type` (with value 0 to indicate subscription), `Server-Name` (with the SIP URI of the SCIM), `Send-Data-Indication` (with value 1 to indicate request of data) and Public-Identity (with the public identity of the desired user).

After the successful processing of a UDR/SNR by the HSS, a UDA/SNA is returned via the Sh interface to the SCIM with the updated details. The Diameter response would contain, among other details, the iFC for that user that relate to services that are composed by the SCIM. The SCIM only gets information about composed services since an IMS application server only has access through the Sh interface to the iFC that pertain to services for which it is responsible for executing. From the set of iFC, it is important for the SCIM to obtain the service URLs (Uniform Resource Locators) for the composed services in order to compile the corresponding composition rules for that service. The iFC are received by the Composition Engine which has the responsibility of creating a vector of service invocation information to be stored in the Composition Rules Cache and used during online interaction management.

The option of storing information that is to be used by the SCIM in the HSS is supported by the design but is not encouraged. The nature of the composition information for complex services is different from that which is currently supported by static interaction mechanisms. It also requires a change in the structure of the service profiles of IMS users, which violates the design goals. The discussion above on the use of Diameter is included because several proposals that were reviewed in Chapter 3 used the HSS for composition information. This discussion has shown that EISL can support this strategy if it is decided in future that composition information should be hosted by the HSS.

Rather, EISL encourages the use of the XDMS and the XCAP protocol for offline interaction management. The procedure of offline interaction management using XCAP is similar to the procedure described above in terms of the involvement of the components in the SCIM. When a

SIP REGISTER is received by the SCIM from a registering user, the SCIM examines the Composition Rules Cache to see if it has a copy of the service invocation vector for that user. If not, it issues an HTTP GET request to the user service repository requesting the service subscriptions for that user. When the SCIM receives the HTTP 200 OK response back from the XDMS, it extracts the information from the response which is in XML format and passes it to the Composition Engine in order for it to generate the service invocation vector for user services. The service invocation vector contains an ordered list of service capabilities that must be executed for each service on the basis of priority. Any service personalisations for each service capability are also appended to the vector. The Composition Engine generates the vector and stores the rules in the Composition Rules Cache.

4.2.4.2 Online Interaction Management

Online interaction management occurs within the context of an ongoing session. As previously explained, the SCIM receives a SIP request from the IMS Core and must then decide how to execute that request. Since users must always register before they can use network resources, and since every user registration results in the auto-generation and storage of composition rules, it is highly likely that composition rules for that user will be available by the time the request is received by the SCIM. In certain circumstances the operator may decide that these records must be removed after a certain time interval in order to conserve memory or storage space. In which case, the SCIM simply performs the actions related in the previous section before continuing to service the user request.

When a SIP request from a registered user is received by the SCIM via its ISC interface, it consults the Composition Rules Cache in order to determine which service capabilities are required to execute that service. By using the cache, the SCIM obtains the identities of the relevant service capabilities, the order of their execution and any preferences and policies that must be followed to influence service delivery in the appropriate manner.

The SCIM obtains the service capability with the highest priority for that service and examines the policies that must be applied for its execution. In EISL, feature interaction policies take precedence over user preferences, so those rules are consulted first. Policies might bar the firing of a service capability or may change the way in which the SCIM invokes it. For this, the SCIM compares the identity of the next service capability to be executed with the previously executed one and ensures that this firing does not violate any feature interaction rules that are specified in the user service repository. Next the SCIM examines the user preference rules against the request in order to determine if the service will behave in the way that is expected by the user. The preferences may bar an interaction from occurring or may require the SCIM to change the way in which it handles the request. If none of the preferences are violated by the firing of the service capability, and any changes that need to be made are made, it sends the SIP request for processing. Once the service capability has performed its function, it returns the request to the SCIM. When the SCIM receives a response back, it inserts the service capability details into an historical vector and loads the information for the next service capability from the invocation vector in the Composition Rules Cache. The SCIM repeats this process for all service capabilities for a service and then returns control to the S-CSCF.

4.2.5 Architectural Placement

The centralised SCIM is the placement that is easiest to describe, which is why the SCIM is represented in a centralised position in the EISL solution architecture. From the perspective of the S-CSCF, the SCIM is nothing more than another SIP application server, and from the perspective of the application servers, the SCIM is the S-CSCF. One design option that has been adopted by other researchers is to incorporate the SCIM into the S-CSCF. In EISL, the SCIM is a standalone agent and is not a part of the S-CSCF. This design decision ensures that the SCIM is as pluggable as possible, and does not modify the structure of existing nodes like the S-CSCF. This is therefore considered to be a better approach.

EISL lends itself to other architectural placements besides the centralised version. The distributed model is also implicitly supported. This architectural placement model requires that multiple SCIM nodes be deployed and that the S-CSCF route requests to the appropriate SCIM. By assigning users to different SCIM nodes through configurations in their service profiles, distribution can be realised as the S-CSCF uses the iFC in the service profile to determine the target of the request. Similarly, the hybrid model can be realised through similar configurations.

In addition to this, in order to support the distributed and hybrid models, SCIM nodes must be able to communicate with each other. In this case, the same procedures for offline interaction management apply except that each SCIM must have its own third party registration so that the S-CSCF routes the request to the specific SCIM serving that specific user. An EISL SCIM is well suited for interacting with an application server since, by definition, it is able to perform interaction management for various application servers that host their own service capabilities and are spread throughout the network. Each SCIM must be able to interact separately with the HSS to download user data and request notifications of changes in this data, which all EISL

SCIMs are able to do. This design even scales well because as the number of users increases, the interactions required for the execution of their services are shared among a number of SCIM nodes. This holds if the allocation of users to SCIMs is performed in such a way that each SCIM handles interactions for more or less the same number of users as the others. Since both the centralised and distributed architectural placements are catered for in EISL, then the hybrid placement is also implicitly supported.

4.2.6 User Equipment Behaviour and Service Personalisation

The goal of the design in terms of the UE as with other aspects of the existing IMS nodes is to retain standard features and functions so that no significant changes are required. Thus in EISL, the UE communicates in the standard way with the IMS Core, which is via the SIP Gm interface. The design capitalises on the standard requirement that the UE must be in possession of an XCAP stack which provides the Ut interface for service management purposes [43]. Using the embedded XCAP stack, the UE downloads service composition information and also submits composition amendments to the user service repository.

The ability to download service information from the network to the UE is a novel and important aspect of the design. Currently, it is not possible for the UE to discover and obtain information about subscribed services since that information is normally only accessible to the S-CSCF and to application servers through the Cx and Sh interfaces respectively. This information is therefore not normally available to the UE since the UE does not have a Diameter interface with the HSS. When service information is recorded in an XDMS, then the UE has direct access to such information. If that information can be downloaded and presented to the user in a way that is intuitive and easy to understand, then the user will be able to compose services in a way that better reflects their preferences at that time.

Since service information is contained in the user service repository, the UE would download the information from there. To obtain the desired information, the UE sends an HTTP GET request to the user service repository. The XDMS would then simply respond with an HTTP 200 (OK) response, with the service information attached. In some cases, the user service repository has no information to present to the user. This would happen if the user has never personalised her services. The correct behaviour in this case would be for the XDMS to send an HTTP 404 (Not Found) response back to the UE. Upon receipt of this response, the UE would need to download the default service composition information from the global service repository. The HTTP GET request would be resubmitted to the service repository, which would respond with an HTTP 200 (OK) message with service information attached.

Once the information has been returned, the user would be able to browse compositions on their terminal and subsequently edit them according to their own preferences. When the updated service information is ready, the user can upload personalised composition rules to the user service repository with an HTTP PUT request. If there were no records in the user service repository when the request was submitted, the XDMS would confirm the creation of the new file with an HTTP 201 (Created) response. If the upload constitutes a modification of existing records, the XDMS responds with an HTTP 200 (OK) response. This scenario is depicted in Figure 4.3.

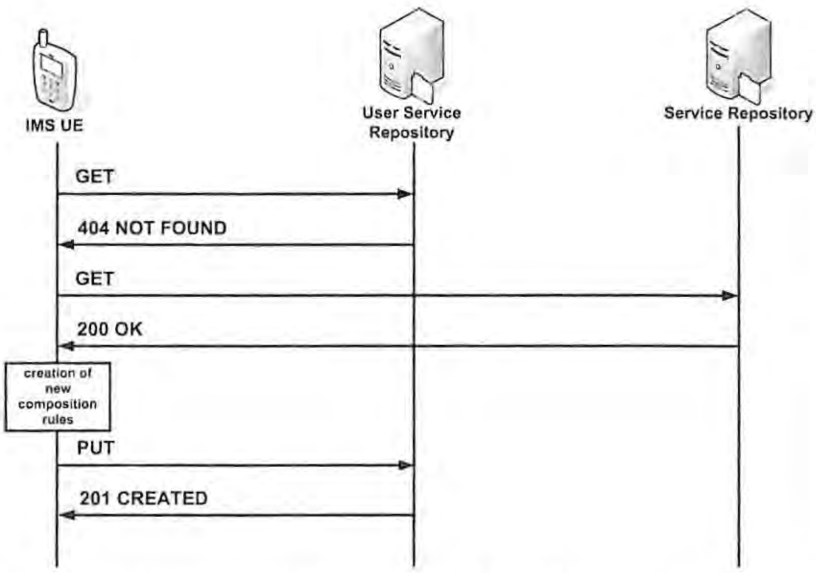


Figure 4.3: Interaction between the UE and the user service repository: Option 1.

One of the benefits of the XCAP protocol is that it can be used by low-powered devices on bandwidth-constricted connections. It realises this goal in that once a file has been created on the XDMS, the client device need only send small fragments of information in the form of document selectors across the access network [67]. In contrast, the elements in the IMS Core are typically powerful machines with a significant amount of bandwidth resources at their disposal. From Figure 4.3, it is evident that the UE must be able to respond to an HTTP 404 (Not Found) response and re-submit the request to the service repository. The additional two messages that result from his condition and the intelligence that is required on the end device to handle this case are not cause for much concern. However, it is possible to eliminate the need for such interventions in the UE by introducing the SCIM as an XCAP proxy that sits between the UE and the EISL repositories. Figure 4.4 shows this alternative scenario.

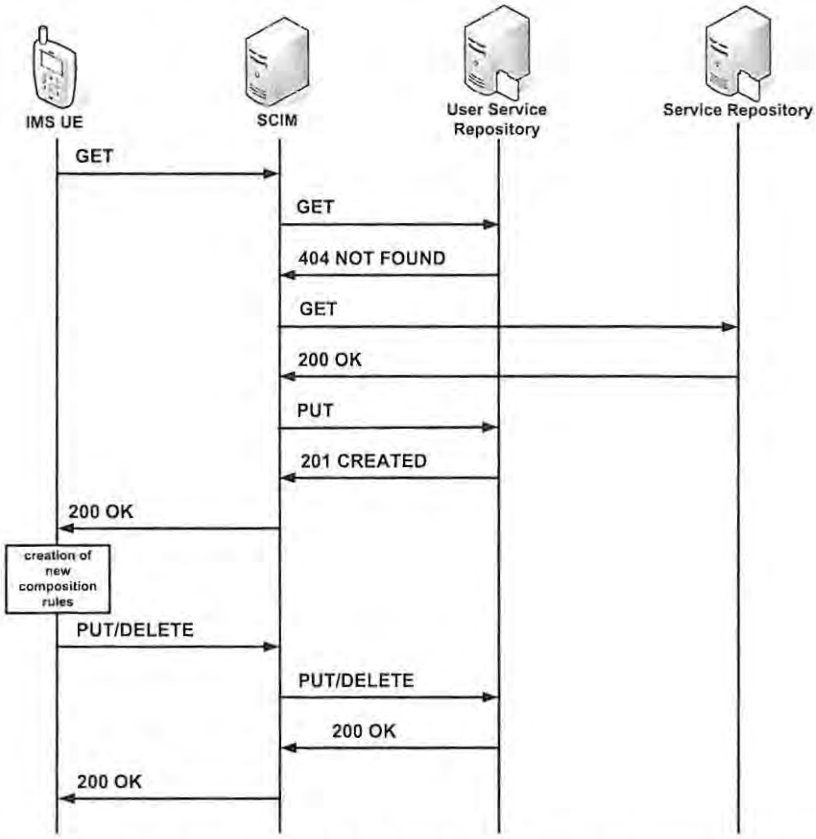


Figure 4.4: Interaction between the UE and the user service repository: Option 2.

While it is clear that the total number of messages is increased using this alternative approach, there are fewer messages that are transmitted over the access network, which ultimately conserves bandwidth. The benefit of this strategy is much more obvious when there are thousands or tens of thousands of customers being served on the same access network. This strategy also imposes fewer requirements on the UE software since it does not need to handle the additional HTTP 404 (Not Found) response itself, nor does it need to be configured with an additional XDMS server address to query. So, while both approaches are supported by the design, the second option is the one that is recommended.

4.3 Discussion

This chapter has described the design of an extended IMS service layer, or EISL, that addresses the requirements of service decomposition, feature interactions, service interaction management and service personalisation. The major design considerations that influenced the specification of EISL are the close alignment to existing standards and the use of existing IMS protocols and procedures wherever possible. Where standards have not kept up with the requirements, the design uses commonly used IMS technologies and interfaces in order to realise new functions and to enable new behaviours. This is exemplified by the re-use of common protocols such as SIP, Diameter and XCAP in various parts of the design. Another closely related objective is the minimisation of the overall impact of introducing the SCIM into the IMS service layer. This is realised by the pluggability of the SCIM. The introduction of the SCIM places no additional requirements on the existing service architecture and all entities maintain their existing interfaces in order to communicate with the SCIM. This is an important objective, as it could impact substantially on the appeal of the SCIM to network operators, and ensuring the pluggability property could reduce the cost of deploying the SCIM in live IMS networks. Instead of requiring changes in the way the IMS network is configured or behaves, there are conceptual changes that are made that fit in with the existing infrastructure. For example, the design allows existing static interaction management configurations to continue, but allows for services which the operator has flagged as composed services to be directed towards the SCIM as the designated application server. Thus the design eliminates the need to specify actual application servers in the iFC by routing requests for composed services to the SCIM, which assumes the responsibility of handing the requisite service interactions. The flexibility of the design is further reflected in its ability to cater for centralised, distributed and hybrid architectural placements of the SCIM. Though the discussion is largely made in reference to the centralised SCIM, it is apparent that due to standard mechanisms that partition resources such as the IMS subscriber base among several servers, the introduction of more than one SCIM node can be done without great impact, and also scales well. User devices can also interact with the new service repositories using its existing XCAP interface, requiring only that the user provide the configuration URL for the XDMS. All in all, EISL provides a plausible solution to the challenges that have been discussed in this thesis. The next chapter details the first step towards an implementation of the concepts that have been described. It shows how open source software can be used to develop IMS application servers and demonstrates a multimedia service that was developed using a communication platform called Mobicents that adds value to a prototypical IMS network.

Chapter 5

IP Multimedia Services in Practice

From the literature that has been reviewed so far, two things are evident. The first is that the IMS Core and its associated application layer are complex. Various mechanisms and protocols from both the Internet and the telecommunication domains must be merged in order to deliver multimedia services across a diverse array of access networks and devices. The second is that SDOs such as 3GPP, 3GPP2 and ETSI are mainly concerned with developing technical specifications that detail the structural and functional architecture of the IMS subsystem and are not directly involved in the actual development of prototypes that can help verify the robustness and suitability of those specifications. Proof of concept implementations have an important role to play in this regard, providing a critical feedback path to the standardisation process, and can inform operators and service providers alike on how effective these mechanisms are at achieving their objectives in real world scenarios. The freedom to experiment with IMS services is predicated on the availability of an IMS environment and the presence of services that can be used to interact with it. The challenge of the complexity of the IMS can be overcome by providing developers with the tools necessary to engage in open experimentation. Though proprietary solutions do exist on the market, many of these solutions are out of reach to the average developer. As a cheap and viable alternative, open source software can offer the developer what is needed to gain such experience. As a result, a testbed environment that mimics the behaviour of a real IMS network would be of great assistance to an IMS researcher. This chapter looks at the practical side of developing and deploying services in an IMS testbed, beginning with a look at the Open IMS Core project, which is a free and open source implementation of the IMS Core. An examination of some of the most popular software platforms in use for developing services in an open source testbed is presented and their suitability as standards-based IMS application servers is assessed.

As a prelude to the development and testing of a the SCIM in an open source testing ground, a system that was developed from the codebase of an open source telecommunication platform called Mobicents is presented, which shows how a single software platform can be used to deploy multimedia services for IMS.

5.1 FOKUS Fraunhofer Open IMS Core

The Open IMS Core [68] is an open source software project which began at the end of 2006 and is managed by the FOKUS Fraunhofer research institute in Germany. Among its competence centres, FOKUS has one called the Next Generation Network Infrastructures (NGNI) centre that provides expertise on innovative implementations of multimedia services over converged, heterogeneous networks [69]. NGNI is the centre primarily responsible for bringing the Open IMS Core project to the world.

The Open IMS Core is sometimes referred to as the *Open IMS Playground* because in addition to being open source and IMS-centric, it is regarded as an environment in which developers and other stakeholders can "play" with the latest IMS technologies. For this reason, it can be considered the central part of an IMS testbed - a test laboratory that lends itself to developments and extensions. The Open IMS Core has continued to grow and mature since its inception, mainly through input from academia and commercial partners, and offers support both to novice and advanced users through its active mailing list discussions. As its name suggests, the project is an implementation of only the core IMS nodes which are the CSCFs and the HSS.

5.1.1 Open IMS Core: Call Session Control Functions

Prior to the creation of the project, FOKUS Fraunhofer supported an open source software project known as SIP Express Router, or SER. SER became part of a set of multimedia software projects under the collective banner of iptel.org that also included the SIP Express Media Server (SEMS) and the SIP Express Router Web (SERWeb) [70]. SER is developed using the C programming language and provides the basic functionalities of a SIP registrar, SIP proxy and SIP redirect server. These default behaviours can be extended by adding new functions to the server. Each function is implemented through a software module that can be configured using module-specific parameters that determine the server's behaviour. What makes SER popular is that these modules can be used to extend the basic SER functions to provide more advanced features such as persistent storage (through MySQL, Oracle and Postgres database integration), load balancing, NAT traversal, presence and availability, messaging and many more.

The Open IMS Core CSCFs are derived from the SER codebase and are implemented as SER modules [68]. There is also a Diameter module called `CDiameterPeer` that provides the CSCFs with the Cx and Dx interfaces to interact with the HSS. An additional ISC module is also provided that allows the S-CSCF to interface with a SIP application server [71]. Listing 5.1 shows an extract from a working S-CSCF configuration file. The syntax of the configuration is relatively easy to understand and uses English-like statements. The `loadmodule` command is used to load a binary module and the `modparam` command is used to set the module parameter values that are consumed by the binary module. Lines 5, 9 and 12 in the file show the loading of the S-CSCF, ISC and Diameter features respectively. The modparam statements that follow specify the configuration of those functions. The listing shows part of the logic for handling a SIP registration request (lines 15 to 17) and the issuing of a security challenge to the registering UE (lines 21 to 25).

Listing 5.1: Example of S-CSCF configuration module.

```
1  listen=146.231.122.16
2  port=6060
3  alias="scscf.open-ims.test":6060
4
5  loadmodule "/home/mtsietsi/OpenIMSCore/ser_ims/modules/scscf/scscf.so"
6  modparam("scscf","name","sip:scscf.open-ims.test:6060")
7  modparam("scscf","user_data_xsd","/home/mtsietsi/OpenIMSCore/ser_ims/modules
       /scscf/CxDataType_Rel7.xsd")
8
9  loadmodule "/home/mtsietsi/OpenIMSCore/ser_ims/modules/isc/isc.so"
10 modparam("isc","my_uri","scscf.open-ims.test:6060")
11
12 loadmodule "/home/mtsietsi/OpenIMSCore/ser_ims/modules/cdp/cdp.so"
13 modparam("cdp","config_file","/home/mtsietsi/OpenIMSCore/scscf.xml")
14
15 route{
16       if(method=="REGISTER"){
17             route(REGISTER);
18       }
19 }
20
```

```
route[REGISTER]{
    if(!S_is_authorized("open-ims.test")){
        S_challenge("open-ims.test");
        t_reply("401","Unauthorized - Challenging the UE");
        exit;
    }
...
}
```

5.1.2 Open IMS Core: Home Subscriber Server

The FOKUS Home Subscriber Server (FHoSS), unlike the CSCFs, is developed in the Java programming language [72]. At its core, FHoSS uses the `JavaDiameterPeer` stack. It is an implementation of the Diameter base protocol and is a sister project to `CDiameterPeer`. The actual data itself, such as user profiles, are stored in a relational database and FHoSS uses the Java Hibernate framework to abstract from the actual database implementation. Hibernate provides an object/relational mapping function to create mappings between code objects and tables in a relational database by using metadata descriptions [73]. FHoSS uses the Apache web application framework known as Struts in combination with servlet technology to provide a web-based management interface that supports functions such as the insertion of IMS users and the specification of services.

Listing 5.2 shows how the `CDiameterPeer` module of the S-CSCF can be configured to interact with the HSS. Line 7 shows the fully qualified domain name and port number of the FHoSS and line 9 shows the IP address and port number of the S-CSCF.

Listing 5.2: Example of a Diameter peer configuration for the FHoSS.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <DiameterPeer
3      FQDN="scscf.open-ims.test"
4      Realm="open-ims.test"
5          ...
6
7      <Peer FQDN="hss.open-ims.test" Realm="open-ims.test" port="3868"/>
8
9          <Acceptor port="3870" bind="146.231.122.16"/>
10         <Auth id="16777216" vendor="10415"/><!-- 3GPP Cx -->
```

```
        <Auth id="16777216" vendor="4491"/><!-- CableLabs Cx -->
        <Auth id="16777216" vendor="13019"/><!-- ETSI/TISPAN Cx -->

    <DefaultRoute FQDN="hss.open-ims.test" metric="10"/>
</DiameterPeer>
```

5.2 Service Platforms for the Open IMS Core

Free and open source software tools for telecommunciations have an important role to play in providing a developer with practical experience with IMS protocols [74]. In this section, an overview of some of the free and open source software platforms that are available for developing IMS services is presented. The selection of the projects under review was influenced by their popularity as reflected by papers that detail service integrations in IMS testbeds, some of which are mentioned later in this section. Discussions on the Open IMS Core mailing list and the author's own personal interactions with researchers in this area also contributed to this discussion. The objective of the investigation was to identify platforms that could provide the functionality of an application server or that of an MRF for media services. With regard to application servers, the scope of the investigation was limited to SIP application servers alone, since from the perspective of the IMS Core and the SCIM, the IM-SSF and OSA SCS servers are essentially SIP application servers. The various types of application server behaviours such as originating/terminating UA and 3PCC were also considered. Regarding media services, the platforms were assessed on their ability to provide the required media processing and media control functions.

5.2.1 SIP Application Servers

5.2.1.1 Asterisk

Asterisk is an open source IP PBX (Private Branch Exchange) written in the C programming language and is released under a General Public License (GPL) version 2 [75]. Its name is derived from both the key on the standard telephone and the Linux wildcard symbol [76]. Asterisk is a multi-protocol application server, but can function as a SIP server through its SIP stack. Asterisk also doubles as a robust media server that provides media services such as audio conferencing, voicemail, announcements, call agent services, CDR and others.

Service creation in Asterisk follows one of four paths. The most basic of these is the Asterisk dialplan. The dialplan is a text-based language that works by listing the legal set of numbers that can be dialed by a subscriber followed by an action to be taken. That action is usually the invocation of a service. Asterisk defines several built-in services that can be invoked through the dialplan. The Asterisk Gateway Interface (AGI) is another service approach that works by extending the basic dialplan by referencing custom programs instead of standard Asterisk services. AGI uses scripts that can be developed in almost any higher level language such as Java, Perl, C, C++, Python and others. A third mechanism is through the use of text files called Call Me files that can be dropped into a standard directory on the server that is running Asterisk. These files are either executed immediately or are spooled for execution at a later date. Each file contains a set of commands for Asterisk to execute. Lastly, Asterisk defines an API called the manager API that allows external programs to communicate with it over a TCP/IP connection.

Asterisk can behave as both an originating and terminating UA. The techniques described above facilitate these roles. Asterisk can also behave as a SIP proxy through the use of the dialplan. It can also behave as a 3PCC. However, Asterisk fails as a standard application server for IMS because it does not have a Diameter stack to provide the Sh interface. There is evidence of ongoing work to add Diameter functionality to it, as reflected by the active issue #0015006 [77] which resulted in a patch for Asterisk to add CDR support using Diameter. However, this alone is inadequate for the requirements of the Sh interface for an IMS SIP application server.

5.2.1.2 OpenSER (Kamailio / OpenSIPS) / SER / SIP Router

OpenSER is an umbrella term for two SIP servers. The first SIP server, Kamailio, is free and open source and is released under a GPL version 2 license [78]. When it made its first appearance in 2005, it was forked from SER and was released under the name OpenSER, but due to trademark reasons, the name was changed to Kamailio in 2008. The second SIP server, OpenSIPS, is also free and open source. It also splintered from SER and is very similar in structure to both Kamailio and SER. More recently, Kamailio and SER developers have combined efforts to create a new project called SIP Router [79]. This project began in late 2008 and has unified the two projects under one umbrella, though Kamailio and SER development still continues independently. The term xSER is often used to refer to these servers collectively, since they are derived from the same codebase.

The manipulation of xSER servers is accomplished through the use of a configuration file. The syntax of the configuration file is similar in all three systems and resembles Listing 5.1. Like the CSCF configuration file, the behaviour of the servers is defined using pluggable modules that can be configured with custom values. All three servers provide SIP capabilities by default and import additional modules for performing SIP-based operations such as UE registration and presence event handling. xSER servers also provide AAA functions using Diameter modules. Though Diameter modules do exist, the type of Diameter support does not correspond with the requirements of the Sh interface. As a result, xSER servers cannot be used in their current state as full SIP application servers for IMS. It is only the separate effort that was undertaken to produce the Open IMS Core CSCFs that was able to provide the required Diameter functionality to an evolved version of SER.

With regard to application server modes, xSER servers can provide all the roles that are defined. User agent client modules help provide the UA capabilities. Stateful and stateless SIP proxy modules are available. SIP dialog modules also provide the requisite dialog management operations to enable the 3PCC functions.

Developers in [80] showed how a presence server could be developed for IMS using a combination of OpenSIPS and an open source XDMS called OpenXCAP [81]. OpenSIPS is particularly well suited to interoperate with OpenXCAP and provides a special module called OpenSIPS-mi-proxy in addition to a native XML-RPC module to interact with OpenXCAP.

5.2.1.3 SIP Servlets (Sailfin / Mobicents SIP Servlets)

SIP servlets are defined in JSR [1] (Java Specification Request) 116 (version 1.0) [82] and JSR 289 (version 1.1) [83]. The SIP servlet specification defines a container-based approach, similar to the HTTP servlet specification, for developing SIP applications. A SIP servlet is an application that is managed by a container environment and performs SIP signaling. The container that hosts the servlets can be an extension to an application server that provides the network services for handling SIP requests and responses [84]. The revision to the initial servlet specification (version 1.1) defines extensions to features that were defined in the original specification (version 1.0). These extensions were provided as feedback from industry partners. Among the improvements, application composition features were refined to define when, if and how a servlet application is invoked in respect to another. Application invocation features were also refined. JSR 116 defined an ordering for the triggering of rules within the individual applications, however the refinement

[1] A JSR is a Java technical specification. They are developed through the Java Community Process which is an open process that allows interested parties to take part in the development and extension of the Java programming language.

in version 1.1 would allow the definition of behaviours that would determine the order in which the triggering rules for servlet applications are considered. Application convergence was refined to move seamlessly between HTTP and SIP servlets in a converged application.

Sailfin is an open source SIP servlet container that is now under the Oracle suite of products [85]. Servlets developed using the Sailfin framework are deployed onto a Java enterprise server called Glashfish. Sailfin provides native SIP support and an Sh interface for interacting with an HSS. In addition to this, it also provides the Ro and Rf interfaces that are required to handle online and offline billing in IMS [86].

Mobicents SIP Servlets, a sub-project of a Java communication platform called Mobicents is another JSR 289 compliant SIP servlet container [87]. It can be deployed on either a JBoss application server or on a Tomcat web server. The container provides Sh interface support through functionality that can be imported from the Mobicents Diameter Server [88]. The Diameter server provides Cx and Dx interfaces as well as the Credit Control Application, which is an IETF application that uses Diameter to implement realtime credit control [89]. Both servlet containers, through the use of their SIP capabilities and Diameter interfaces, fulfill the requirements of a SIP application server for IMS.

Developers in [90] describe the use of Sailfin as an application server that was part of a testbed called IMS Innovation. The purpose of the testbed was to simplify the development of services for developers with little knowledge of SIP by exposing simple APIs for them to use. In an experiment, the authors were able to develop a testbed that enabled web developers to create a web application that monitored the activities of an athlete using Sailfin as a converged web/SIP application server. The server acted as a SIP watcher for a user's heart rate readings and published them to a web page.

5.2.1.4 Mobicents JAIN SLEE

JAIN SLEE (Java APIs for Intelligent Networks) (Service Logic and Execution Environment) is a Java standard described in JSR 240 [91] that defines an event-based programming model, application lifecycle and management facilities for diverse communication services. As a Java standard, JAIN SLEE is a product of the JCP (Java Community Process) which allows interested companies and private individuals to participate in the evolution of the Java platform and its future extensions. JAIN itself is an umbrella term for a set of Java protocol stacks such as SIP, MGCP and others [92, 93]. The support for multiple Internet protocols allows JAIN SLEE to interoperate with larger systems that are also based on open protocols [94].

SLEE leverages the abstractions provided by JAIN by defining a framework for the development of resource adaptors, or RAs. These adaptors reside outside of the SLEE environment, but perform the important task of adapting external protocol messages (such as SIP messages) into events that can be consumed by the SLEE and vice versa. For service creation, JAIN SLEE utilises the concept of service building blocks (SBBs) to define atomic, reusable application components that can subscribe to receive events and generate responses to those events. This function is aided by an event router that is responsible for routing events to SBBs that register their interest in receiving certain events. At the time of writing, Mobicents is the first and only open source certified implementation of the JAIN SLEE 1.0 and 1.1 standards and is released under an LGPL license [95]. As with Mobicents SIP Servlets, Mobicents SLEE SBBs can also import Diameter functionality from the Diameter server to implement an Sh interface with an HSS.

Researchers in [96] describe a framework called SPICE, which hosts Mobicents SBBs, J2EE components and web services. The SLEE is used to convert SIP calls into web service requests and J2EE remote method invocation calls.

Table 5.1 shows a summary and comparison of the feature sets of the four application service platforms under review. All of the application servers can execute the defined application server roles, but only the SIP servlet engines and JAIN SLEE provide the additional Sh interface to interact with the HSS.

	Term UA	Orig UA	SIP Proxy	3PCC	Sh
Asterisk	✓	✓	✓	✓	✗
xSER	✓	✓	✓	✓	✗
SIP Servlets	✓	✓	✓	✓	✓
JAIN SLEE	✓	✓	✓	✓	✓

Table 5.1: Comparison of SIP application server platforms.

5.2.2 Media Resource Function

5.2.2.1 Asterisk

As previously explained, Asterisk is both a service platform and a media server. Asterisk can provide the Mr interface of the MRF through its SIP stack. However, Asterisk is designed to be a "one box solution" and although most of the elements in the IMS Core are logical entities that can be hosted on a single host, the MRF was intended to be a physically decomposed multimedia server according to the ITU [97]. Moreover, Asterisk does not have H.248 support to control an MRFP. It does however have an MGCP stack. MGCP, or Media Gateway Control Protocol, is described in RFC 2705 and is used by a call controller to control a VoIP gateway [98]. However, the MGCP function in Asterisk can only be used in call control mode, and Asterisk itself cannot be controlled by another media server via MGCP. Asterisk provides very robust support for media codecs as it supports the G.711a, G.711u, G.722, G.723.1, G.726, G.729, GSM, ILBC and speex audio codecs in addition to H.261, H.263, H.263p and H.264 video codecs.

Asterisk has not been used widely with the Open IMS Core by the research community. In [80], researchers used Asterisk as a PSTN gateway. Asterisk supports analogue telephony hardware through drivers that have been built specifically with Asterisk in mind, and can be used to terminate calls from the Open IMS Core to the PSTN or legacy mobile networks. Asterisk has also been used in [99] in conjunction with OpenSER to connect an IP PBX network with an Open IMS Core testbed.

5.2.2.2 SIP Express Media Server (SEMS)

SIP Express Media Server (SEMS) is a media and application platform for VoIP services based on SIP and is released under a dual GPL version 2 license and a proprietary license [100]. SEMS is not used in standalone mode, but requires the availability of a SIP server to peer it with. As an iptel.org project, SEMS has been extensively tested with SER and the project provides documentation for how SEMS can be used in conjunction with SER to provide media services. Like its sister project, SEMS is based on a modular design that defines basic functionality, and supports the insertion of plugin modules that add extra functionality. A core API based on C++ is available for extending the capabilities of the server, but developers can also use an embedded Python interpreter to implement extensions as Python scripts. SEMS performs basic media services such as the playback of announcements, voicemail and conferencing. For this, SEMS support the G.711a, G.711u, iLBC, GSM and speex audio codecs.

5.2.2.3 Mobicents Media Server

The Mobicents Media Server is the media component of the Mobicents communication platform and is released under an LGPL version 2.1 license [101]. It is an open source media server, developed in the Java programming language and supports both IP and legacy interfaces. As such, it can act as both a media server and a media gateway. Mobicents Media Server provides a SIP interface which it can use to implement the Mr interface with the S-CSCF. It also defines several call control protocols for controlling media processors. These include MGCP, MEGACO, MSCML and JSR 309. MEGACO is an IETF protocol defined for controlling a media server, and is compatible with H.248 [102]. MSCML, or Media Server Control Markup Language, is also an IETF protocol and defines a markup language that can be used in conjunction with SIP to control a media server. MSCML is usually used for media conference services [103]. JSR 309 defines a generic, easy to use media control API that hides the complexity of the underlying call control protocols from the developer [104]. This API only works with JSR 309 compliant media servers. Mobicents Media Server supports the G.711a, G.711u, GSM, G.729 and speex audio codecs, in addition to the H.261 video codec.

Table 5.2 shows a summary and comparison of the feature sets of the media systems under review. Asterisk can be used for call control and media processing in general IP networks, but since the call control protocol in IMS for an MRFC is H.248 and not MGCP, it does not fulfill the requirements. Neither does Asterisk decouple its call control from its media capabilities. The same analysis applies for SEMS. Only the Mobicents Media Server provides all the functions required for an MRF. All necessary control protocols are supported, media processing abilities are catered for and the call control and media processing functions are decoupled.

	Call Control	**Media Processing**	**De-coupled**	**Mr**	**Mp**
Asterisk	✓ MGCP and SIP	✓	✗	✓	✗
SEMS	✓ SIP	✓	✗	✓	✗
Mobicents Media Server	✓ MGCP, SIP, H.248, JSR 309 and MSCML	✓	✓	✓	✓

Table 5.2: Comparison of media resource function platforms.

5.2.3 Integrating Application and Media Services

Some of the software platforms that have been described offer a portion, but not all of the required features for application and media services in IMS. For example, Sailfin offers only SIP application server capabilities and must therefore be used in conjunction with a media server to provide media services. Because one platform may not always provide all the necessary features, the developer is forced in certain cases to use different platforms to provide the necessary functions. Platforms such as Asterisk and Mobicents provide both application and media services at the same time and try to offer a consistent development platform for the provision of all types of services. From the discussion above, it is evident that only one platform, Mobicents, is equipped to provide both application and media services.

The other decision the developer must make is whether to implement all features on a single host, or to spread the features across several physical hosts. Both approaches are possible as the IMS technical specifications define logical roles, and there is nothing that prohibits all nodes from being deployed on a single host [2]. It is intuitive that deploying multiple functions on a single node is more convenient than using several nodes as it eases both the integration and management of those services. However, this deployment choice suffers from the dangers associated with a single point of failure which is undesirable. If the functions are spread across multiple nodes, a single application platform can still be used, as it too eases the integration and management of services. This approach seems to be beneficial both in the short term (for the installation of the system) and in the long term (for the management of that system over a long period of time, and over numerous software upgrades).

Due to the sheer scope of the project, Mobicents is able to satisfy most of the requirements for service delivery in IMS. Either the JAIN SLEE server loaded with a SIP RA or the SIP servlet container can provide the functionality for the ISC interface. The Mobicents Diameter stack can be used to implement the Sh interface and the Mobicents Media Server can be used as an MRF, with all the call control and media processing functions that are required.

In [16], the author attempted to address the challenge of selecting service platforms for an IMS testbed. Given that some of the platforms that have been described can only be used to provide a subset of the required features, the author acknowledged that there is a temptation to import different components into the testbed and to only use each one for a specific purpose. This is possible as long as the different pieces fit in with each other and proper interfaces can be provided. This opportunity is evident in collaborations such as the one that exists between OpenXCAP and OpenSIPS or SEMS and SER, where special care has been taken by the developers to ensure compatibility and interoperability. While these components may be able to interoperate on a functional level, there are disadvantages that are associated with a heterogeneous testbed that are the inevitable result of these kinds of integrations.

In a heterogeneous testbed, the developer must be aware of the full set of software dependencies prior to installation. It would be convenient if all of the service platforms had common dependencies, but this is not always the case. For instance, xSER servers including their plugin modules, have the same set of dependencies (except for the XML RPC libraries) in order to communicate

[2]The MRF is an exception to this rule, as it must be physically decomposed.

with OpenXCAP. However OpenXCAP has its own set of dependencies as well. Therefore, to deploy a presence service that includes an XDMS, several dependencies are needed. The installation itself is a trivial task, but these libraries need to be maintained and the software needs to be kept up to date with operating system upgrade schedules defined by the system administrator. Conflicts can easily arise in such situations.

Programming languages will also differ. For instance, xSER servers, SEMS and Asterisk are all developed in C. Knowledge of C is required for developing new modules or modifying existing ones. OpenXCAP uses Python and Mobicents SBBs are implemented in Java. The use of multiple service platforms places a high demand on the skill set of testbed developers. Even when the same programming language is used, the configuration syntaxes may still differ. xSER servers have a similar syntax, but SEMS has its own syntax, as does Asterisk, and so on. Skills in one platform are not transferable to another in these cases. For a competent developer, these hurdles need not be prohibitive, but requires extra effort to extend and maintain such installations. As the testbed expands, administrative demands can become burdensome and time consuming. Table 5.3 summarises the software dependencies that some of the software platforms have.

	Kamailio SIP AS	**Kamailio PS**	**Kamailio RLS**	**OpenXCAP**	**Asterisk**
Development language	C	C	C	Python	C
Configuration	Kamailio scriptlet	Kamailio scriptlet	Kamailio scriptlet	OpenXCAP scriptlet	Dialplan
Pre-requisites	gcc, bison, flex, make	gcc, bison, flex, make	gcc, bison, flex, make, libxml2, libxmlrpc-c3, libxmlrpc-c3-dev	libxml2, python, python-twistd, python-application, python-gnutls, python-lxml, python-twistd-web2, python-zopeinterface, python-mysqldb	libc6, libcurl3, libgcc1, libncurses, libnewt, libssl, zlib1g

Table 5.3: Dependencies in a heterogeneous IMS service layer. Source: [16].

Figure 5.1 shows the possible design of a heterogeneous testbed that uses only those components that qualify for integration with an IMS testbed as identified in the previous sections. Services can be built using Sailfin, Mobicents SIP Servlets or Mobicents JAIN SLEE. The conspicuous presence of Mobicents components is evident in the figure. By eliminating Sailfin from the picture, the testbed would consist of only Mobicents components which could be achieved without losing any functionality. If the developer desires to deploy all services on one physical node, that option is possible by running a single instance of JBoss and loading all the required components into it. Components can also be separated by running multiple JBoss instances on a single machine. It is also possible to run single instances on different machines as well. Whatever deployment option is chosen, the end result is the utilisation of a single development platform to deliver multimedia services, with the added benefit of creating a more manageable service layer. It is more manageable because there is a single common dependency, which is the Java runtime environment. Also, it requires a scoped set of skills (namely the Java programming language) and allows the transfer of skills across the different Mobicents sub-projects, which use a common configuration language to describe projects, Maven.

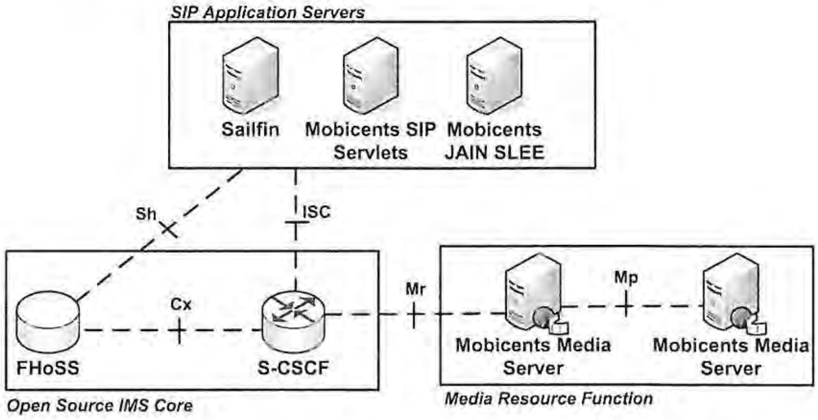


Figure 5.1: A heterogeneous testbed for IMS services.

Recognising the ability of Mobicents to deliver all the required multimedia services of an IMS testbed has led to it being chosen as a reference implementation for the design that was presented in the previous chapter. In the next section, the implementation of a multimedia service that combines web and SIP services to enable a converged video shop over an IMS testbed is described. The system was developed by the author and three other postgraduate students. It serves as a precursor to the use of Mobicents to implement a SCIM for IMS and was used to gain exposure to the Mobicents service model.

5.3 Mobicents Converged Demo with Video Support: An Example of IMS Service Development and Deployment

The Mobicents communication platform consists of five distinct sub-projects. Mobicents JAIN SLEE server is an implementation of JSR 240. Mobicents Diameter server provides a Diameter stack for AAA functions and IMS interfaces. Mobicents SIP Presence service provides a presence server, an XDMS and a resource list server that implement IETF, OMA, and 3GPP standards. Mobicents SIP Servlets provides a JSR 289 compliant SIP servlet container and Mobicents Media Server is a media server for VoIP and IMS media applications. Each sub-project can be downloaded with a bundled JBoss application server and pre-loaded binaries for easy running. The binaries implement example services that show the capabilities of the platform. For educational purposes, and to communicate the programming model to novice developers, the source code for the sub-projects and the corresponding example programs are available for download via subversion and as tarred packages. Figure 5.2 shows the system layout of Mobicents.

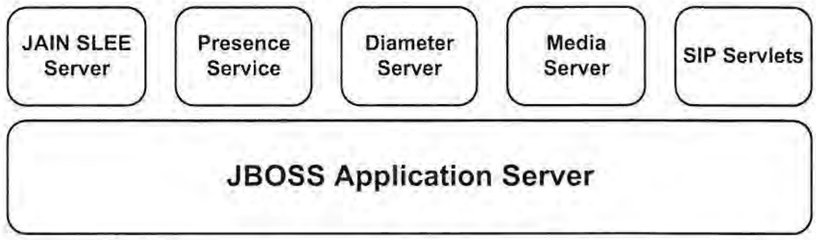


Figure 5.2: Mobicents system layout.

One such example is the Mobicents Converged Demo which is bundled with the Mobicents Media Server [105]. The Converged Demo combines SIP and TTS (text-to-speech) resource adaptors as well as media functionality, with the JBoss Seam web framework. The web framework implements a web system that allows users to create accounts on it. Each account maps a username with a SIP address. When the user logs in, the system authenticates the user and loads the corresponding SIP address into memory for use at a later stage. Once authenticated, the user then proceeds to browse through a catalog of furniture items. As the user shops, items can be loaded into a shopping basket. The user can view the items in the shopping basket at any point, add new items, remove existing items and view the resulting sub-totals.

When a user has finished shopping, she places an order for the items that have been selected. Once the order has been placed, a SIP call is automatically initiated to the user's SIP terminal. When the user answers the call, the TTS engine plays an announcement requesting the user to key in the desired delivery date for the order using their keypad. Once this has been accomplished, the call terminates and the order is sent to the e-commerce part of the system, which implements a backend that stores information about orders that are yet to be processed. The order will then appear under a 'My Orders' tab in the web system. As an aside, the Converged Demo requires an administrator to confirm each order before a call is made to the user, but these details are outside the scope of this discussion.

This example shows how Mobicents can be used to create an interactive service application that combines application logic based on SIP with media features. It also shows that Mobicents can be integrated with JEE technologies (i.e. JBoss Seam) to add multimedia services to enterprise software.

After testing the system and recognising the possibilities that could be exploited, the author and three colleagues endeavored to use the Converged Demo as the basis for an extended version of the service. The idea was to use the basic functionality and code to create a system that allowed registered users to browse through a web catalogue of videos instead of furniture items. Users would be able to view information about the videos on offer, including plot summaries and prices, from their browsers. If a user wanted to view a trailer of one of the videos before actually buying it, she would simply click a button in the web page and the system would initiate a call to the SIP address associated with the registered account. Upon answering, the TTS engine of the media server would be invoked to play an announcement that would prompt the user to confirm the request to have the trailer streamed to their device by pressing the answer key. When the system receives this confirmation, streaming functionality in the media server would be triggered to begin streaming the video to the user's device. After viewing the trailer, the user would be able to add the video to the shopping basket and continue browsing the online catalogue. When ready, the user would go through the ordering process as in the original system, but in this case, the media server would play an announcement that requests the user to confirm or decline the order by using a key press. If confirmed, the user would confirm the delivery date of the video as usual and the order would be placed.

The key benefits of the system, particularly from a service provider perspective, are three-fold. The first is that this version of the Converged Demo introduces video as a new form of media, which represents added value. Secondly, whoever deploys such a system would be able to protect it from abuse by ensuring that only registered users were able to access it. Whoever is browsing the catalogue on the web would have to be in possession of the actual SIP device associated with the account in order to fully enjoy the features of the system. This is particularly important if the system is targeting paying customers only. Thirdly, the system works just as well in an Internet environment as an IMS environment since it is based on the SIP protocol. The service can easily be integrated into an IMS testbed by inserting the relevant iFC configurations to allow integration with the Open IMS Core.

The prospect of deploying the service in an IMS testbed was explored and successfully implemented. Figure 5.3 shows the architectural layout of the system that was developed.

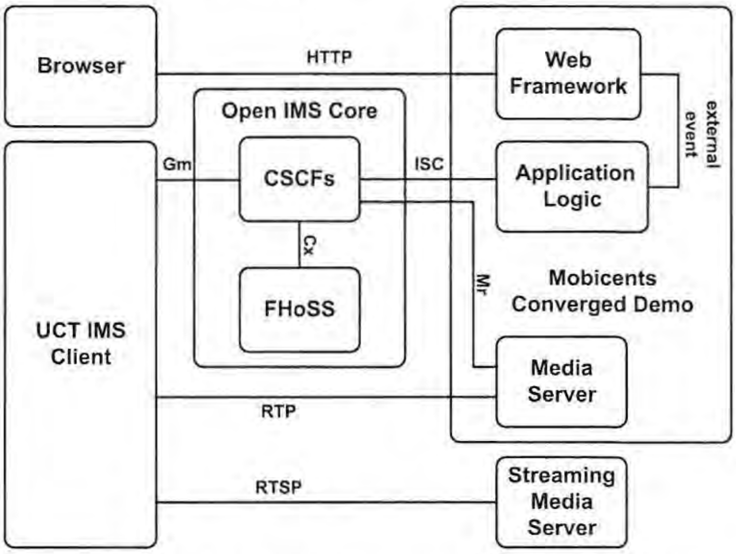


Figure 5.3: Architecture of the Mobicents Converged Demo with video trailer streaming.

The next section discusses issues related to IMS UE software and the following sections describe the interventions in the original code that were necessary to enable this system.

5.3.1 IMS Client

In order to connect to the service in standalone mode, any SIP client can be used. However, to access the service in an IMS environment, an IMS compliant UE is required. In addition, due to the streaming requirements of the system, the UE must have support for an IP-based streaming protocol. 3GPP's technical specification on streaming services states that an IMS UE accessing streaming services must have an RTSP stack to negotiate media delivery with a streaming media server [106].

There are a few IMS clients that are freely available for download on the Internet. Among them is the Mercuro IMS client [107], Monster [108], IMS Communicator [109] and the UCT IMS Client [110]. The most appropriate UE was found to be the UCT IMS Client. There were two reasons for this. Firstly, unlike other clients like Mercuro and Monster, the UCT IMS Client has an RTSP stack. The IMS Communicator embeds JMF (Java Media Framework) which does provide user agents with RTSP functionality, however JMF no longer receives support from Oracle (the company that owns it) and has not had an update to the API since 1999 [111]. JMF also lacks support for many codecs, a problem which has led to the emergence of several Java-based media frameworks such as Jffmpeg [112] and FMJ (Freedom for Media in Java) [113] that offer better codec support. The RTSP stack in the UCT IMS Client is implemented using gstreamer which is a modular, pipeline-based framework for developing media services [114]. The UCT IMS Client was also chosen because it has support for content indirection for SIP. Content indirection and its use in the Converged Demo is explained in Section 5.3.2.3.

5.3.2 Interventions in the Converged Demo Sources

There were several interventions that needed to be made in order to introduce new features into the existing project. This section lists these interventions and describes the technical details behind what was done.

5.3.2.1 Database Entries

Much like FHoSS, the Converged Demo uses the Hibernate framework to map objects to tables in a relational database. The system uses HSQLDB (Hyper Structured Query Language Database). HSQLDB is written in Java and provides applications with a small footprint, embeddable relational database engine. This functionality is used when the system must fetch and present the video list to the user. The values are hardcoded using SQL commands contained in a file named import.sql. When the server is started, the file is automatically executed and the values are loaded into the database. Default user accounts were already available, but new ones could be loaded using a statement like the following:

```
INSERT INTO USERS (USERID, DTYPE, FIRSTNAME, LASTNAME, ADDRESS1, ADDRESS2,
CITY, STATE, ZIP, EMAIL, PHONE, CREDITCARDTYPE, CC_NUM, CC_MONTH, CC_YEAR,
USERNAME, PASSWORD) VALUES (2,'customer','Mosiuoa','Tsietsi','4 Privet Dr
ive','Cupboard under the Stairs','QSDPAGD','SD',24101,'m.tsietsi@mobicent
s.ac.za','sip:mosiuoa@mobicents.ac.za:5070',1,'1979279217775911',03,2012,
'mosiuoa','mosiuoa')
```

The videos that are sold on the system are stored in a database table called PRODUCTS. Some of the previous field names from the furniture store were re-used, but other fields needed to be added as well to reflect the new use case. As such, insert commands for the PRODUCTS table resembled the following:

```
INSERT INTO PRODUCTS (PROD_ID, ASIN, TITLE, PRICE, IMAGE_URL, TRAILER_URL,
DESCRIPTION) VALUES ('5', '11PgsxMDwRL._AA160_', 'The boy in the striped
Pyjamas ', 100.00, '/img/25gh0gm.jpg', 'rtsp://mediaserver.mobicents.ac
.za:8000/channel5', 'Young Bruno lives a wealthy lifestyle in Pre-war
Germany along with his mother, elder sister, and army Commandant father.
The family re-locate to the countryside where his father is assigned to
commandeer a prison camp. A few days later, Bruno befriends another youth,
strangely dressed in striped pyjamas, named Shmuel who lives behind an
electrified fence. Bruno will soon find out that he is not permitted to
befriend his new friend as he is a Jew, and that the neighboring yard is
actually a prison camp for Jews awaiting extermination.');
```

The SQL command contains a new field called TRAILER_URL. This URL provides the RTSP address for the video trailer. This address, as will be shown later, is used by a SLEE SBB to let the UE know the RTSP address for the trailer that is requested.

5.3.2.2 Custom SLEE events

The SLEE container is well suited to handle external events from outside the SLEE such as SIP messages from a SIP phone. These messages are adapted by resource adaptors to generate SLEE events that can then be appropriately routed by the SLEE event router to an event consumer. However, SLEE also supports custom events that are not generated by entities such as external protocol stacks, but are defined by the developer. The Converged Demo uses this custom event support to fire events towards the SLEE that are consumed by Converged Demo SBBs. So, when a user clicks on a link in the web page to play a trailer, the Seam framework fires a custom event that was developed specifically for trailer support called OrderTrailer which is consumed by an SBB called UserSBB object. The incorporation of the new event was added by creating the following event configuration:

```
<event event-direction="Receive" initial-event="True">
<event-name>OrderTrailer</event-name>
    <event-type-ref>
        <event-type-name>
            org.mobicents.slee.service.dvddemo.ORDER_TRAILER
        </event-type-name>
        <event-type-vendor>org.mobicents</event-type-vendor>
        <event-type-version>1.0</event-type-version>
    </event-type-ref>
<initial-event-select variable="ActivityContext" />
</event>
```

The event bundles customer information that is extracted from the session, so that the `UserSBB` object can be made aware of important information such as the customer's name and the RTSP address of the trailer that is being requested. This custom event definition requires the creation of an event handler in the `UserSBB` object to handle the request. The SBB uses the media API to interact with the TTS resource adaptor to generate an announcement that is played to the user. Figure 5.4 shows the routing of SIP messages as SLEE events to SBBs and the routing of custom events from Seam to the SBBs.

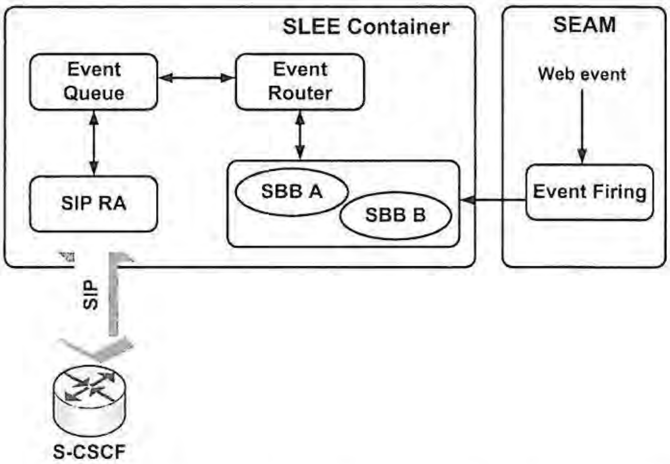


Figure 5.4: Firing and receiving of events in the Converged Demo.

5.3.2.3 Content Indirection

The user must confirm the reception of the trailer by pressing the call button on their IMS Client. The call button is used to generate a SIP reINVITE, which is an in-dialog INVITE that is normally used in SIP to change the multimedia properties of a SIP session. Not all UEs support this function, but the UCT IMS Client does which is another reason why it was chosen for the experiment. The iFC must be configured to allow the SIP reINVITE to be sent to the Mobicents container. This was effected in the FHoSS. Once the reINVITE is received by the application server, it uses content indirection to redirect the SIP request to an RTSP address. Content indirection for SIP is defined in RFC 4483 which allows a MIME type in a SIP message to be referred to indirectly via a URL [115]. In this case, the application server responds to the reINVITE by sending a SIP 200 (OK) response to the UE with the following ContentType header:

```
ContentType: message/external-body; access-type="URL"; expiration="Sat,
01 September 2009 09:09:34 GMT"; URL="rtsp://mobicents.ac.za:8000/
channel5"
```

The ContentType header above specifies the MIME type as *message/external body*. An expiry date is also provided in addition to the RTSP address for the location of the trailer on the streaming server. Once the UCT IMS Client receives this message, it automatically strips this information from the SIP message, and uses its RTSP stack to submit the request for the video stream.

5.3.2.4 Streaming Server

RTSP streaming in Mobicents was introduced in September 2009. It is based on the Netty framework and uses NIO Socket support [116]. In the initial phases of the experiment, RTSP support in Mobicents was still in its infancy, as such it was decided to use an external streaming server. At first, Darwin, the open source streaming server developed by Apple was used. In a latter phase, VLC was used to provide the streaming capabilities. In the final stage of the experiment, Mobicents RTSP functionality was used.

In each case, the UCT IMS Client contacts the streaming server with an RTSP DESCRIBE request which retrieves the description of a media object by way of a URL. In response, the server responds with an RTSP 200 (OK) response message that contains media initialisation information in the SDP portion of the message. The UE uses this information to construct a SETUP request indicating the transport parameters to be used for streaming, with the server responding to each SETUP request with an RTSP 200 (OK) that includes a session identifier. The PLAY request from the UE tells the server to commence delivery of media via the mechanism specified in the SETUP request. The PLAY request contains the URL extracted from the SIP 200 (OK) sent from Mobicents. The media is delivered as RTP packets with video data in the payload. The user can terminate the RTSP session with a TEARDOWN request. The usual behaviours of the system continue from this point. The user can continue shopping or proceeds to the checkout system. Since the original system requires inband DTMF support and the UCT IMS Client only uses outband DTMF via the SIP OPTIONS message, a new handler was created in the `UserSBB` class and iFC were provisioned to ensure proper routing to the application.

5.3.3 Security and Standards Conformance

The new version of the system takes advantage of specific functionality in the UCT IMS Client such as RTSP and content indirection. The developers of the UCT IMS Client also developed an IPTV (IP Television) application server that uses the same content indirection model to play video channels [117]. By their own admission, this model of IPTV suffers from a security flaw in that RTSP addresses could be extracted from SIP messages received by the UE. This would allow the UE to send RTSP requests to the streaming server without logging into the system. Also, the use of content indirection is not the standards-based way of performing IPTV functions in IMS. 3GPP and TISPAN have specified IPTV architectures for IMS in [106] and [118] respectively. These technical specifications describe an architecture that supports personalised content discovery and filtering and the delivery of electronic program guides that provide information regarding available channels. The author has subsequently explored issues related to IPTV service discovery and personalisation in [119].

5.3.4 Converged Demo Without IMS

A screencast of the system was developed that shows how the system works. It was posted on YouTube in November 2009 [120] and to date, has registered over 600 views. The same components that have been described so far were used, but instead, the system was deployed without the Open IMS Core. The protocols that are used in the system, such as SIP, RTP and RTSP, are protocols that were developed for the open Internet and work just as well without

IMS. In order to simplify the setup of the system and to encourage even novice Mobicents users to enjoy the system, IMS configurations were cut out and the system allowed to run in standalone mode. This version of the system used the Darwin streaming server and the JAIN SIP Applet phone as a SIP client. Appendix B provides screenshots of different parts of the system.

5.4 Discussion

This chapter has focused on the practical aspects of experimenting with the IMS through the use of open source software. SIP is a common IP protocol and as such there is a plethora of SIP servers available for download on the Internet. However, IMS has strict requirements which are not always met by existing servers. There is a clear lack of systems that can be used for IMS, and the investigation was only able to identify two suitable platforms for this. In addition, there are media platforms that are used to deliver multimedia content in general IP networks. However, the IMS too places constraints on the platforms that can be chosen for this, as the media resource function must decouple the media control functions from the media processing functions, and also requires a specific media control protocol be used, namely H.264. An architecture was presented that showed that it is possible to use existing open source software to provide multimedia services, and the Mobicents platform was shown to provide most of what is required for this. Interventions in the Mobicents Converged Demo system were described as an example of service development and deployment in an IMS testbed. In the next chapter, Mobicents components are used to develop a SCIM for IMS. Lessons that were learnt from the investigations related in this section as well as from the Converged Demo were important in this step, as a choice needed to be made regarding whether to use Mobicents SIP servlets or Mobicents JAIN SLEE.

Chapter 6

Implementing the Extended Service Layer

The EISL system was presented with the objective of communicating the design decisions that this thesis motivates. The thesis has aimed to present some of the competing strategies for enabling these features by reviewing existing proposals and critically analysing their arguments. In certain cases, either for practical reasons or otherwise, the design decisions that have been adopted have been easy to deduce and justify through this process of analysis. The topic of architectural placement is an example of this. It is intuitive that a central SCIM is less likely to cope under strain than either a distributed or hybrid SCIM. However in other cases, the correct decision has not been straightforward. In such cases it is often necessary to validate the decision based on an implementation of the design. An implementation has been developed for EISL, and therefore the objective of this chapter is to relate the author's experiences in validating the design through the use of the implementation. Naturally, there are several possible realisations of EISL as nothing in the design is tied to specific software platforms or development languages. What is presented here is simply one of the possible realisations. The implementation was influenced by the experiences gained from the previous chapter on the practical aspects of IMS service development and deployment. This chapter carries on from there, showing how the Mobicents communication platform has been able to support the development of an EISL and to model the functional architecture of a SCIM for IMS. The SCIM that is presented is an initial prototype that is meant as a proof of concept.

6.1 Choice of Implementation

The SCIM is a SIP application server and if Mobicents is to be used as a platform for the development of a SCIM, there are two technologies to choose from which support the development of SIP application servers. These are JAIN SLEE and SIP servlets.

Comparisons between SLEE and servlets have been the focus of several studies in the past such as in [121] where the authors evaluate the suitability of the two technologies as application servers for IMS. The authors make their comparison based on factors such as the degree of code re-use, concurrency control, support for multiple protocols, JEE integration and operations and management support. In their conclusion, the authors state that SIP servlets are good at developing simple SIP applications but that JAIN SLEE provides a more sophisticated solution to satisfy the requirements of an IMS application server.

In another paper by Bessler *et al*, the authors describe the combination of SLEE with a workflow engine using BPEL to control SBBs and external entities [122]. Here, the authors also choose SLEE as a service platform, citing SLEE's powerful internal event model and the added value that is derived from being able to integrate other protocols besides SIP as motivating factors.

Ivelin Ivanov, Director of Product Development at JBoss and one of the core Mobicents system architects also provides some insight into this argument in a web article where he compares the Mobicents implementations of the two technologies [123]. Ivelin notes the relationship between SIP servlets and JEE and finds that the servlet model is an easier programming model than JEE. He also mentions that Java developers not familiar with JEE are likely to be comfortable with SIP servlets. Due to the fact that servlets in general are derived from a common `GenericServlet` class, developers familiar with HTTP servlets are also likely to find SIP servlets easy to use. Among the criticisms that Ivanov levels against SIP servlets is that they do not define a rigid component model that separates call control from the business logic, unlike SLEE and other Java containers. In his article, he describes JAIN SLEE as a powerful environment, but one that is very complex and presents a steep learning curve for developers.

In choosing a platform, the developer must focus on what is being developed and the system requirements. JAIN SLEE provides power and high configurability and also supports multiple IP protocols. However this benefit comes at a high cost due to its complexity. The SCIM presented in EISL is a SIP application server needing only SIP, Diameter and XCAP interfaces and has no need of many of the RAs that JAIN offers. Later in this chapter, the relatively straightforward process of including Diameter and XCAP support for SIP servlets without the use of RAs will be shown. The complexity of SLEE is a clear deterrent for modeling a SCIM for an experimental testbed. Previously, the author has supported the use of a SLEE server to provide the SCIM function citing the powerful SBB composition model and low latency properties of SLEE as attractive features [124, 125]. This holds for service capabilities that are hosted on the same application sever as the SCIM and are implemented as SLEE SBBs, however, the SCIM is likely to be responsible for composing external service capabilities. Thus the power and low latency properties of SLEE are never fully realised since external enablers must be invoked and not SBBs that are internal to the SLEE.

6.2 Experimental Setup

The testbed that was created for the series of experiments related in this chapter consists of two main parts: the Open IMS Core and Mobicents. The Open IMS Core was setup and installed on a desktop computer that runs the Ubuntu Linux 10.10 LTS operating system with an Intel Core i7 processor and 2 x 2048MB RAM. Open IMS Core provides three main download options. These is subversion source code access, daily snapshots of the source code that are compiled on a nightly basis and a ready-to-run virtual machine based on the Gentoo Linux operating system. Subversion provides an easy way of downloading the project components as well as receiving and reviewing changes to files that have been checked in. The daily snapshots are provided as a convenience to the user since they do not require the installation or configuration of a subversion client. The virtual machine image is the most convenient of all the options since it spares the user the effort of having to build the components. It does however require the user to work in a specific flavour of Linux, which is Gentoo.

Having considered the different options, it was decided that one of the code snapshots would be used since there was little need to keep track of changes to the IMS components. A snapshot from December 2010 was chosen and utilised in the testbed. Open IMS Core also requires the installation of a database and a DNS server. MySQL 5.1 server was chosen as a database system and Linux BIND9 as a DNS server. The domain name open-ims.test was used for the testbed. The DNS zone file for this domain is provided in Appendix C.

Mobicents also offers different download options to developers. The code for SIP servlets can be downloaded through subversion or as a tarred binary package. The latter was chosen and the Mobicents SIP Servlets 1.4.0 FINAL package was downloaded from the Internet. The servlet container was then installed and run on the same machine as the Open IMS Core. To supply the

Diameter functionality, the Mobicents Diameter Server 1.3.1 FINAL binary package was used.

The Mobicents SIP Presence Integrated BETA 6 binary package was used as an XDMS. This integrated version provides both SIP presence and XDMS features in one download. The deployment options are such that the presence and XDMS servers can be run together or the XDMS can be run in standalone mode. Currently, the presence server cannot be run in standalone mode. Since presence features were not required, the XDMS was run by itself. To run it, an HP EliteBook 6930p laptop was used that runs the Ubuntu Linux 10.10 LTS operating system with an Intel Core 2 Duo processor and 2048MB RAM.

In order to mimic the service capabilities that the SCIM interacts with, the Kamailio SIP server was used. Version 3.0.0 of Kamailio was chosen and installed on the desktop computer. Multiple instances were run to simulate multiple service capabilities.

6.3 Setting up Mobicents for Integration with Open IMS Core: FHoSS Configurations

To enable communication between the Open IMS Core and Mobicents configurations can be made in the FHoSS web console. This section describes these configurations, including the structure of the trigger points that support both offline and online interaction management, and shows how IMS users can be assigned to a service profile that allows their requests to be forwarded to the SCIM.

The Open IMS Core can be provided with details regarding the SCIM that help the S-CSCF locate and communicate with it over the ISC interface. The configurations that enabled these requirements are shown in Figure 6.1. The SCIM is given the fully qualified domain name of mobicents.open-ims.test which is also reflected in the DNS zone file. The Open IMS Core can also be informed of the items of information in the HSS that the SCIM is allowed to access using the Sh interface. This is important if certain information needed for interaction management is stored in the HSS. Though EISL supports this, it is not the recommended approach and the implementation provided in this chapter does not use the Diameter Sh interface. As such these configurations are not important. The iFC that are linked to the SCIM, `mobicents_ifc` and `register_ifc`, are also listed in Figure 6.1. The trigger point contents of these iFC are described next.

Third party registration support is easy to provide and only requires the creation of a trigger

Figure 6.1: Application server configuration for the SCIM.

point that specifies that SIP REGISTER requests from users should be sent to the SCIM[1]. Figure 6.2 shows the trigger point configuration that enabled this. The trigger point is given the name `register_tp`. There is no need to specify actual SIP IMPUs since by definition the SCIM is bound to a single HSS and handles all service requests for complex services originating from users on the same HSS. This satisfies the case of a centralised SCIM, but even if the SCIM function is distributed, this property still holds since each SCIM is allocated a portion of the user base through iFC configurations that point to different SCIMs for different users.

Live requests from users for complex services must be forwarded to the SCIM by the S-CSCF. Figure 6.3 shows how support for this was achieved by combining three SPTs that formed the trigger point named `mobicents_tp`. The first SPT matches a SIP INVITE, the second matches a request URI and the last matches an originating session request. The second SPT uses pattern matching to match any SIP request URI that contains the domain name open-ims.test.

Once the iFC had been set up, a service profile was created that would allow users on that profile to have their communication handled by the SCIM. A service profile was created with the name `mobicents_sp` and its configuration is shown in Figure 6.4. This service profile was allocated

[1]The alternative would be to use the Diameter Sh interface to subscribe for IMS User State changes.

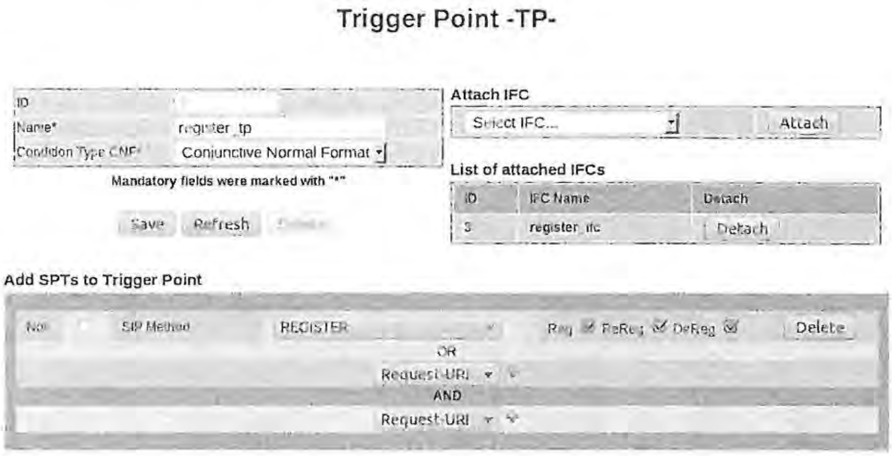


Figure 6.2: Third party registration configuration for the SCIM.

to the user profiles of two IMS users, Alice and Bob, which were used in the series of experiments that are discussed in this chapter.

Figure 6.4: Service profile for users served by the SCIM.

6.4 SCIM Configuration Repository

In the EISL design, the SCIM is required to have a Configuration Repository that contains information that the SCIM needs to operate on the network. Though this repository can be used for any type of information that the operator identifies, there are a few items of information that are particularly important to the SCIM. These are details regarding the HSS, the XDMS and the

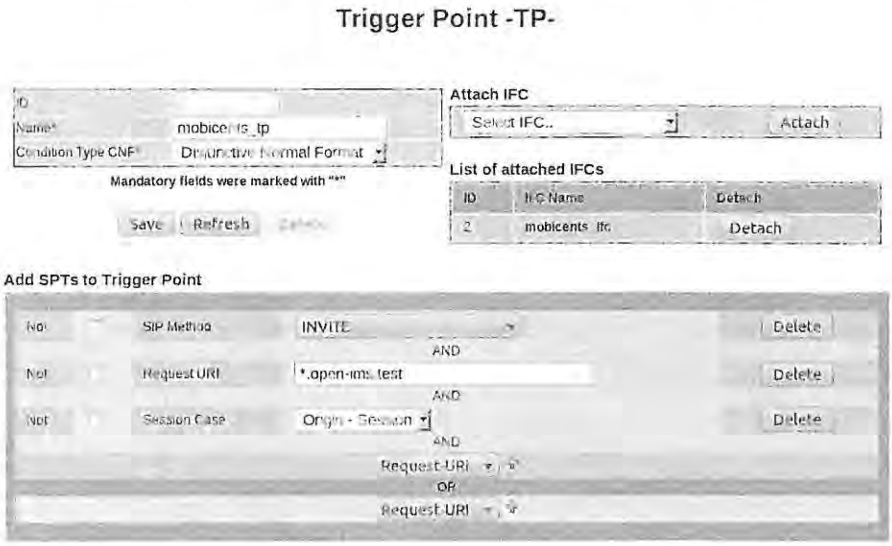


Figure 6.3: SIP INVITE configuration for the SCIM.

service capabilities. HTTP servlets make use of a web.xml file as a deployment descriptor for the configuration of the servlet. Similarly, SIP servlet applications use a sip.xml file for the same purpose. It is possible to use the sip.xml file to provide servlet applications with the required information. According to the syntax of the file, `<context-param>` elements can be used and the actual information be provided using the `<param-name>` and `<param-value>` elements. An example that shows how this was done for the XDMS is given in Listing 6.1. Similar configurations for the service capabilities are provided later in this chapter.

Since the context parameter element supports the creation of custom parameters, other details that easily conform to a key-value format can be provided in this file. When the servlet application needs to import these parameters into memory, Mobicents provides an API to interrogate this file and allocate variables to store the values. Mobicents also allows applications to create these parameters during runtime. The dynamic creation of information that assists in interaction management and other related processes are covered in the next section that shows how offline interaction management was performed using SIP servlets in Mobicents.

6.5 Offline Interaction Management

The offline interaction management stage is instrumental in providing information to the SCIM that assists it in executing a user request during the online stage. In EISL, the SCIM obtains

```
<context-param>
        <param-name>xdms.provider.url</param-name>
        <param-value>jnp://146.231.121.151:1099</param-value>
</context-param>
<context-param>
        <param-name>xdms.ip.address</param-name>
        <param-value>146.231.121.151</param-value>
</context-param>
<context-param>
        <param-name>xdms.port</param-name>
        <param-value>8080</param-value>
</context-param>
<context-param>
        <param-name>xdms.xcap.root</param-name>
        <param-value>/mobicents</param-value>
</context-param>
```

Listing 6.1: Deployment descriptor example for SCIM configuration details.

information about the users it serves from the SIP REGISTER requests it receives. To model this behaviour, a SIP servlet called `OfflineServlet` was developed and provided with an event handler called `doRegister` that handles REGISTER requests. In this event handler, the servlet extracts the user identity from the From header in the SIP message. The SIP REGISTER request it receives may either be for a new registration or a re-registration. In the case of a new registration the SCIM would not have downloaded the user's service list when it receives the message. It would have to retrieve it from the XDMS.

To download the service list, the SCIM issues an HTTP GET request towards the XDMS. When the SCIM receives the HTTP response from the XDMS, it uses its Composition Engine to convert the raw XML document that is attached to the response into a service invocation vector that stores the services in order of priority. In an experiment, the Java `HashMap` data structure was used to store the list in the form `<Priority, ServiceName>`. In order to share this information with other servlets, the `OfflineServlet` stores the data structure in its context environment. This allows other servlet applications to read this information in the same way that they would read information from the static sip.xml file. The `OfflineServlet` removes this information from its context when the user de-registers. Also, in the event of a new registration the servlet creates an empty history vector that will be used to record invoked services during the live session.

In the case of a re-registration the SCIM simply queries the Composition Rules Cache to ensure that the information is still available. Operators may want older records to expire in order to make room for new, active users. As such the information may be removed without the user actually de-registering. If this is true, the SCIM just downloads the rules from the XDMS. If the rules are still available the SCIM performs no further actions.

Figure 6.5[2] summarises the process of offline interaction management. In an experiment, the Monster IMS client was used as a registering UE to represent offline interaction management on behalf of the user Bob. Bob's account was bound to the service profile `mobicents_sp` in order to correctly invoke the servlet application.

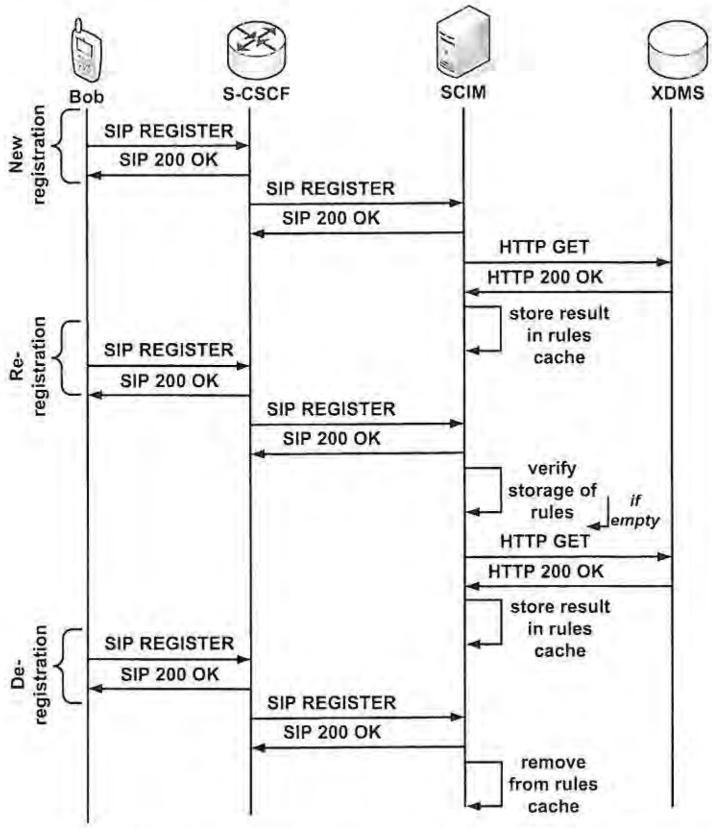


Figure 6.5: Offline interaction management with SIP Servlets.

6.6 Creating and Managing the Service Repositories

EISL uses an XDMS to provide the repositories. When using this technology, it is necessary to identify a suitable XCAP appusage that will govern the structure of the information that will be

[2]The SIP signaling in Figure 6.5 has been simplified for the sake of brevity.

hosted on the XDMS. Currently, there is no appusage that is explicitly defined for the storage and management of information related to service capabilities. However as shown in Section 2.1.4, complex services can be constructed by grouping supplementary services. Supplementary services are comparable to service capabilities since they are not complex and only modify the behaviour of a basic telecommunication service.

TISPAN has defined an appusage for services known as simulation services which are similar in nature to supplementary services. Simulation services are the result of an effort by ETSI to provide fixed line access to IMS services and compatibility between IMS and PSTN/ISDN networks since these types of services have traditionally been deployed on these networks in the past. They include services such as call barring, call forwarding, call diversion, conferencing and others. This appusage has been used in the testbed to represent service composition for IMS services.

6.6.1 Utilisation of ETSI Simulation Services AUID

TISPAN has registered the AUID

$$simservs.ngn.etsi.org$$

with the default namespace of

$$http://uri.etsi.org/ngn/params/xml/simservs/xcap$$

and mimetype

$$application/simservs+xml$$

for simulation services.

In collaboration with TISPAN, 3GPP has specified details regarding the deployment and use of simulation services for IMS in TS 24.173 [126]. In TS 24.623 [13], 3GPP also provides the XML schema for the simservs AUID that specifies the procedures for manipulating simulation service data using XCAP. Figure 6.6 shows how a simulation services document is structured.

The simservs AUID supports the inclusion of one or more supplementary services in a single document. TS 24.623 specifies the XML schema for the common parts of a simservs document

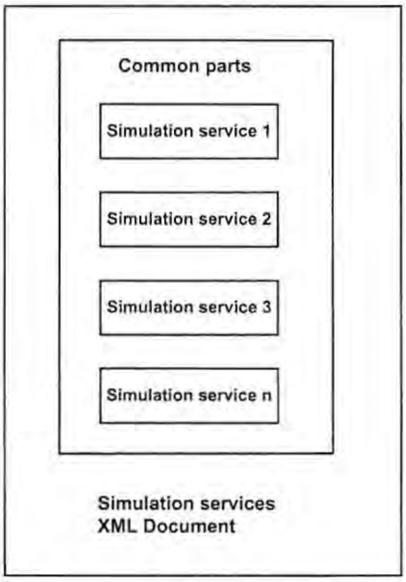


Figure 6.6: Simulation services document structure. Source: [13]

in addition to an XML template for the individual simulation services themselves. The simulation services are located in the same target namespace as the appusage schema. By inserting multiple simulation services into a document it is possible to compose complex services. As an example, the multimedia telephony service specified in TS 22173 [127] is composed of simulation services such as call barring, call diversion and call holding. These work in concert to deliver the complex service. Extensions such as applying an execution order to the simulation services can also be handled when they are recorded in this way since the schema allows elements that are not defined in it to be included. This model thus fits the requirements of the service repositories and makes it a suitable platform for building composed services for IMS.

6.6.2 Concepts Related to the Service Repositories

The service repositories in EISL are required to host two types of information: one is information on the full set of services on offer and the other is personalised user service information. XCAP is well suited to handle both cases since it defines two types of documents, global documents and user documents.

A global document is the parent document for an appusage. It is hosted inside the global tree of all global documents of an XDMS. The global document for the simservs appusage is accessible via a URL of the form

$$http://example.com/mobicents/simservs/global/index,$$

which maps to a global simservs document called index. The mobicents part of the URL is referred to as the XCAP root which is the context for all documents across all appusages on the server.

A user document is hosted in a user's home directory on the XDMS. This document is accessible via a URL of the form

$$http://example.com/mobicents/simservs/users/sip:bob@example.com/index,$$

which maps to a user simservs document called index which resides under a home directory belonging to Bob.

Global documents and user documents have their own authorisation policies. By default, users are only allowed rights to access and modify files in their own directories. Global documents are not normally accessible to ordinary users. The reason for this is that global documents have the potential to affect all user documents of the same appusage through resource dependency rules. These rules may require that a change in a global document reflect in all associated user documents. Due to their sensitivity and scope, only privileged user accounts are usually allowed to access and modify them [42].

Section 2.1.2 provided a description of the types of operations that can be performed on simulation services by a network provider, service provider or service subscriber. The actions of service provisioning, activation, deactivation and withdrawal are performed on the actual service itself. However, these actions could also be followed by a subsequent change in the information stored against the service in either the global simservs document or in a user document. For example, service provisioning, which is the physical action of deploying a new service into the network, could be accompanied by the insertion of the service's details into the global simservs document. This would allow global access of information pertaining to this service. Service activation changes the physical status of a service to an active state. This change in state could also be reflected in the service repositories by allowing the service to only be used when the status is set to active.

Erasure and interrogation are the only operations that explicitly have to do with service data and not the service itself. Global erasure could be enforced by an operator or provider by removing a service from the global simservs document, which would remove it from access to all users.

Erasure could also affect users on an individual basis such that a service is removed from individual users' documents without affecting others. Interrogation allows access to information about the status of services. Service subscribers could interrogate their own user documents to discover the status of a subscription to a service. Also, a subscriber could interrogate the global simservs document in order to discover the status or availability of a service. If the service exists and is available, the subscriber could, for example, issue a request for the discovered service to be added to their user document.

An important feature of the simservs appusage is that it defines the use of policies that can be applied to simulation services. It does so by importing policy schemas that have been defined by the IETF and the OMA. IETF defines a framework called common policy in RFC 4745 that is used to specify policies that can be used to control access to application-specific data [128]. IETF common policy provides an XML schema that describes how these authorisation policies can be applied. The RFC relates mainly to location and presence data, but also states that it could be extended into other application domains.

The IETF common policy schema defines rules that consist of conditions, actions and transformations. A condition constitutes the `if` part of a rule and can either refer to an identity (i.e. a SIP URI) or a sphere (i.e. a state such as work, meeting, home or travel). Validity periods for conditions can also be defined. An action or a transformation constitutes the `then` part of a rule and is enforced when a condition is correctly matched. Actions and transformations are used to permit certain operations to proceed. An action may specify that a presence server must deliver the presence status of a presentity to a watcher, whereas a transformation can tell a presence server to return a modified form of the presence information to the watcher. They cannot, however, be used to explicitly deny an operation to proceed. OMA common policy is derived from IETF common policy and defines an additional but optional set of conditions that are not present in IETF's version [129]. So, in addition to the `<identity>` condition in IETF common policy, OMA defines `<external-list>`, `<anonymous-request>` and `<other-identity>` elements. The external list element matches identities that are part of a URI list. The anonymous request element matches anonymous incoming requests. The other identity element matches all identities and is used as a default policy.

The integration of policy information through the IETF and OMA common policy frameworks into the simservs appusage can allow a SCIM acting on a service repository to apply operator and user policies to the execution of services. In particular, in addition to the default condition elements specified by IETF and OMA, simservs defines extensions to the common policy frameworks by adding new conditions that can be applied. For example, the `<presence-status>`

element is introduced which influences the execution of a service when the presence status of the user is set to a specific value such as available or busy. The `<media>` condition affects the delivery of the service based on the media type that is requested. Other conditional elements include not registered, not reachable and roaming.

6.6.3 Adding a New AUID into the Mobicents XDMS

Having identified a suitable appusage for building the service repositories, the next step was to incorporate it into Mobicents. The Mobicents XDMS supports existing IETF and OMA AUIDs such as resource-lists, common policy, pres-rules and OMA user profile. It also supports the creation and deployment of new AUIDs that are not currently supported. To create a new AUID in Mobicents a new appusage class must be provided that extends the abstract class `AppUsage`. The class `SimServsAppUsage` was created for this purpose. Mobicents provides a number of different constructors for the appusages that allow the developer to specify custom schema validators and authorisation policies. Validators help ensure that a document that is being inserted conforms to the schema and authorisation policies determine if an operation should be allowed or barred. The implementation for the testbed used the default constructor which caused the default behaviours to be adopted in relation to schema validation and the authorisation of XCAP operations.

After the appusage class was created an appusage factory class was required. This class is used to maintain a cache of appusage objects since the validator objects are expensive to create. The class `SimsServsAppUsageFactory` was used as an appusage factory class. It extends the `AppUsageFactory` class defined by Mobicents. This class is used to create a new document when the user requests one to be created.

The third class that was created was the `SimServsAppUsageDeployer` which extends the `AppUsageDeployer` class in Mobicents. This class is responsible for loading and unloading the appusage into and out of the XDMS. The deployer is implemented as a JBoss Microcontainer Bean, which is a means of deploying services in JBoss. The classes were compiled and combined with the deployer file to create a Java archive. The archive was then inserted into the JBoss application server along with simservs schema which completed the installation of the appusage. Figure 6.7 shows a representation of the structure of the archive file that was created. The complete simservs XML schema is provided in Appendix C.

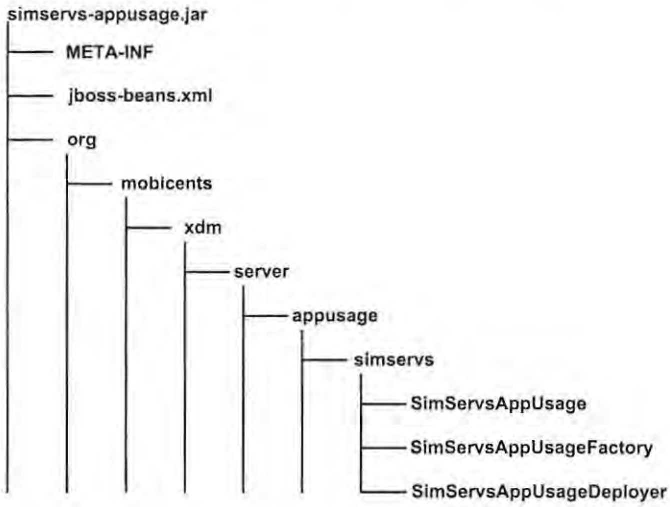


Figure 6.7: Structure of the simservs appusage archive file.

6.6.4 Simulation Services in the Repositories

To demonstrate the use of simulation services and the simservs appusage for interaction management in IMS, four simulation services were chosen as a basis for the service repository. The terminating identity service offers two services, namely the terminating identity presentation (TIP) service and the terminating identification restriction (TIR) service. The TIP service provides the originating party with the possibility of receiving the identity of the terminating party and the TIR service provides the connected party with the possibility of preventing the presentation of the terminating identity to the originating party [130]. The originating identity service also offers two services, which are the originating identity presentation (OIP) service and the originating identity presentation restriction (OIR) service. The OIP service provides the terminating user with the possibility of receiving network-provided identity information of the originating user and the OIR service allows the originating party to restrict the presentation of their identity information to the terminating user [131].

Communication diversion (CDIV) is a service that permits a user to divert communications addressed to them to another destination. CDIV comes in several flavours, such as communication forwarding unconditionally (CFU), communication forwarding on busy (CFB), communication forwarding on no reply (CFNR), communication forwarding on subscriber not reachable (CFNRc) and communication forwarding on not logged in (CFNL) [132]. The communication barring service offers three services, namely the incoming communication barring (ICB) service, the anonymous communication rejection (ACR) service and the outgoing communication barring (OCB) service. The ICB service rejects incoming communications that match certain criteria on behalf of the terminating user. The ACR service is a special case of the ICB service, and rejects incoming communications from anonymous users. The OCB service rejects outgoing communications that fulfill certain criteria on behalf of the originating user [133].

As previously mentioned, the simservs appusage defines its own schema that defines the common parts of the simservs document and allows zero or more simulation services to be inserted into a simservs document. The schema does so by defining a substitution group called `absService`. In XML, substitution groups allow an XML schema to define an element that can be substituted for any one of a set of related elements that name that element as their substitution group. The XML schema for the simservs appusage defines an element called `absService`. In turn, the simulation services listed above each have their own corresponding XML schemas that contain elements that name `absService` as their substitution group. For simplicity, the XML schemas for the simulation services name above were copied into the generic simservs schema and stored as part of the schema for the simservs appusage.

6.6.5 Creating the Global Repository

To implement the service repository, the XML document shown in Listing 6.2 was created and was inserted into the global tree under the simservs appusage on the Mobicents XDMS. The document is plain and does not contain much information except for the names of the services themselves and their default activation values. User policies will eventually be applied during the user personalisation stage of the experiment. By default, the TIP and OIP services do not restrict the presentation of information, a feature that can be changed by a user during personalisation. The global document will be inherited by all users when they begin managing service data for themselves.

```
<?xml version="1.0" encoding="UTF-8"?>
<simservs xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
        xmlns:cp="urn:ietf:params:xml:ns:common-policy"

    <terminating-identity-presentation active="true"/>
    <terminating-identity-presentation-restriction active="true">
        <default-behaviour>presentation-not-restricted</default-behaviour>
    </terminating-identity-presentation-restriction>
```

```
    <originating-identity-presentation active="true"/>
    <originating-identity-presentation-restriction active="true">
        <default-behaviour>presentation-not-restricted</default-behaviour>
    </originating-identity-presentation-restriction>

    <communication-diversion active="true">
    </communication-diversion>

    <incoming-communication-barring active="true">
    </incoming-communication-barring>

    <outgoing-communication-barring active="true">
    </outgoing-communication-barring>

</simservs>
```

Listing 6.2: Structure of the simservs global document.

To simulate the creation of the service repository by the network operator, a SIP servlet called `SimServsGlobalDocumentServlet` was created and provisioned with an insert method that allowed it to issue an HTTP PUT request to insert the global document. Currently the Mobicents XDMS does not define privileged accounts that can be used to perform operations on the global tree. As such, any request to insert a document into the global tree is responded to with an HTTP 403 Forbidden response. However, the default authorisation policy on the Mobicents XDMS is such that local requests to insert documents into the global tree are permitted. This allowed the insertion request to be successfully made by deploying the servlet into a separate JBoss instance on the same machine. Once the global document had been inserted, it could be retrieved from a remote client since the authorisation policy only prohibits the insertion of records from remote clients, and not the reading of them.

User accounts are normally created on the XDMS through the Java Management Extensions (JMX) web console which is provided by the JBoss application server. As an alternative, the Mobicents container provides servlet applications with a remote method invocation (RMI) adaptor to automate this process. An account was created using the `<context-param>` element in the sip.xml file of the servlet. The values contained in the element tag are parsed in the servlet's initialisation method and used to issue a user account creation request to the user profile MBean on JBoss. The username associated with the account was 'sip:mobicents.open-ims.test' and the password was 'mobicents'. These credentials represent those that the operator would issue to the SCIM.

6.7 UE Interactions and Service Personalisation

To test the interactions on the user side, it was necessary to use an XCAP client that could issue XCAP requests to the Mobicents XDMS. Most XCAP clients such as Mercuro and Monster (see Section 5.3.1) have support for XCAP, but have been developed with only a few of the common XCAP appusages in mind, such as resource-lists and rls-services. As such, it is not possible to use these clients to test service personalisation against a simservs document. Therefore, to illustrate service personalisation, a custom SIP servlet called `SimServsUserDocumentServlet` was used to simulate an IMS UE interacting with the XDMS.

For the experiment the servlet was required to personalise the CDIV service acting on the behalf of the user Bob. As previously mentioned, call diversion supports the forwarding of communications to an alternate destination based on provisioned criteria. The servlet would be used to provision such criteria, using conditions and actions defined in the common policy schema.

EISL defines two types of UE interactions with the service repositories: one where the UE is only aware of the user service repository (Case 1) and the other where in addition to this, the UE is also aware of the service repository (Case 2). The use of a custom servlet was particularly important for the Case 2 scenario, since it would be necessary to provide the UE with the URL of the global repository. For the experiment a user document that was derived from the global document was inserted into the user directory. Furthermore, elements and attributes were inserted into the user document and policies were specified.

6.7.1 User Document Creation

In the first step of the experiment, the `SimServsUserDocumentServlet` personalisation application was required to download the currently existing document for the user Bob. A user document selector was constructed and packaged inside an HTTP GET request and was subsequently issued to the XDMS. As expected, the XDMS responded with an HTTP 404 NOT FOUND response as the user Bob did not have a simservs document in his user directory at the time. When this response was received by the servlet it had to issue a second request. The second request was targeted at the global document. A global document selector was constructed and issued to the XDMS and an HTTP 200 OK response was received back with the global document attached as an XML payload. Once retrieved, the contents of the global document were inserted into the user directory using an HTTP PUT request and a user document selector with the URI

$$/mobicents/simservs/users/sip : bob@open - ims.test/index/\ /simservs/index.$$

The mimetype associated with this document is `application/simservs+xml`. As such, in the first stage of personalisation, the status of the user document was exactly the same as the global document shown in Listing 6.2.

6.7.2 Element Creation

In the next step, a new element was to be inserted into the user document. As an example, the user personalisation servlet was used to mimic the action of Bob personalising his CDIV service by requesting CFB functionality to be provided. Based on this information, a telephony service would be expected to forward his calls to a family member named Alice if he were busy. To achieve this, the user personalisation servlet constructs the URI

$$/mobicents/simservs/users/sip : \\ bob@open - ims.test/index/\ /simservs/communication - diversion?xmlns(cp = urn : \\ ietf : params : xml : ns : common - policy)$$

and attaches an XML document to an HTTP PUT request. The request replaces the existing communication diversion element with the one specified in the body of the insertion request. The mimetype associated with this element data is `application/xcap-el+xml`. The query string that appears in the URI above specifies a namespace binding that appears in the document being uploaded. The binding allows for the expansion of the namespace prefix *cp* for the common policy elements. Listing 6.3 shows the content of the CDIV element tag after the insertion.

```
<communication-diversion active="true">
    <cp:ruleset>
        <cp:rule id="CFB">
            <cp:conditions>
                <busy/>
            </cp:conditions>
            <cp:actions>
                <forward-to>
                    <target>sip:alice@open-ims.test</target>
                </forward-to>
            </cp:actions>
        </cp:rule>
```

```
    </cp:ruleset>
</communication-diversion>
```

Listing 6.3: Call diversion XML file with personalisation information.

6.7.3 Attribute Insertion

In the third step of the experiment, an existing attribute belonging to an element in the user document is modified to reflect a user's current preference. One of the possible applications of this feature would be to change the status of a subscribed simulation service. The attribute named active in a simservs element can be switched on or off which corresponds to the activation or deactivation of the service on the part of the user. The user personalisation servlet was used to construct an attribute selector and insert a value of false for the existing CDIV service. The URI for this operation is:

$$/mobicents/simservs/users/sip: bob@open-ims.test/index/\ /simservs/communication-diversion/@active?xmlns(cp = urn:ietf:params:xml:ns:common-policy)$$

which selects the attribute named active in the CDIV element. The servlet attaches the value 'false' to the HTTP PUT request in order to change the active status of the service from true to false. The mimetype associated with this attribute data is `application/xcap-att+xml`. The final state of the document is the same as in Listing 6.3 except that the active status of the CDIV element is changed to false.

These experiments illustrate the creation and insertion of a new user document and the manipulation of document elements and attributes. There are several ways of using the insertion and retrieval operations to modify and interrogate the user document in various ways that have not been demonstrated here. However, the experiments related in this section indicate that by modifying the resource selectors, many of these operations could be performed with relative ease. In the next section, an example of online service interaction management is discussed that uses the properties of a personalised simservs document to influence service delivery for IMS users.

6.8 Online Interaction Management

This section addresses online interaction management and how it was modeled in the testbed using the Mobicents SIP Servlets platform.

6.8.1 The Application Router

In addition to the architecture of the SIP servlet container, the SIP servlet specification also defines an entity known as an application router. The application router is an important agent in the invocation of servlet applications. Though it is not a logical component of the container itself, it is instrumental in the process of choosing which servlet application to invoke in order to handle a request. The application router does not implement application logic, and as such, cannot perform actions such as the modification of a request, or the creation of a response.

The router also has not direct contact with the individual servlet applications, but instead has a contract with the servlet container. When the container receives a request, it passes information about the request to the application router. The application router is allowed to make use of external databases and files that may assist it in making an informed decision about the appropriate application invocation order. The servlet specification also defines a minimalist application router called the default application router (DAR). The DAR is a non-feature rich version of an application router. It simply defines the order in which servlet applications are invoked when a request is received.

The Mobicents SIP Servlets container provides it own implementation of the DAR. To fulfill its duties, the DAR operates on a configuration file that uses a simple syntax that maps a SIP method to a comma separated list of servlet applications and trigger points. An example is given below:

```
INVITE: ("org.mobicents.servlet.sip.testsuite.SimpleApplication",
"DAR:From", "ORIGINATING", "", "NO_ROUTE", "0",
"REGEX=From:.*sip:.*@sip-servlets\.com")
```

From the Mobicents SIP Servlet user guide [87] which is available online, the parts of the file are described as follows:

1. The name of the application as known to the container. The application name can be obtained from the <app-name> element of the sip.xml deployment descriptor of the application, or the @SipApplication annotation.

2. The identity of the subscriber that the DAR returns. The DAR can return any header in the SIP request using the DAR directive DAR:SIP_HEADER. For example, DAR:From would return the SIP URI in the From header. The DAR can alternatively return any string from the SIP request.

3. The routing region, which consists of one of the following strings: ORIGINATING, TERMINATING or NEUTRAL. This information is not currently used by the DAR to make routing decisions.

4. A SIP URI indicating the route as returned by the application router, which can be used to route the request externally. The value may be an empty string.

5. A route modifier, which consists of one of the following strings: ROUTE, ROUTE_BACK or NO_ROUTE. The route modifier is used in conjunction with the route information to route a request externally.

6. A string representing the order in which applications must be invoked (starts at 0). The string is removed later on in the routing process, and substituted with the order positions of sip-router-info data.

7. An optional string that contains Mobicents-specific parameters. Currently, only the DIRECTION and REGEX parameters are supported.

6.8.2 Modeling Online Interaction Management

During online interaction management when a user request is received, the SCIM examines the contents of the message in order to determine the identity of the user that made the request. Once obtained the SCIM queries the Composition Rules Cache in order to determine if the service invocation list for this user has been generated and stored. If not, it downloads the user's service information from the XDMS and generates the invocation list in runtime using its Composition Engine.

The evaluation of the policies that must be applied is the responsibility of the indivdual services themselves. For example, in [133], the ICB service is required to be capable of parsing the common policy rules attached to a user in order to determine how to respond to the request. Therefore, in the case of simulation services the evaluation of policies is not the prerogative of the SCIM but of the individual services that will be executed. Each time, the SCIM proxies the

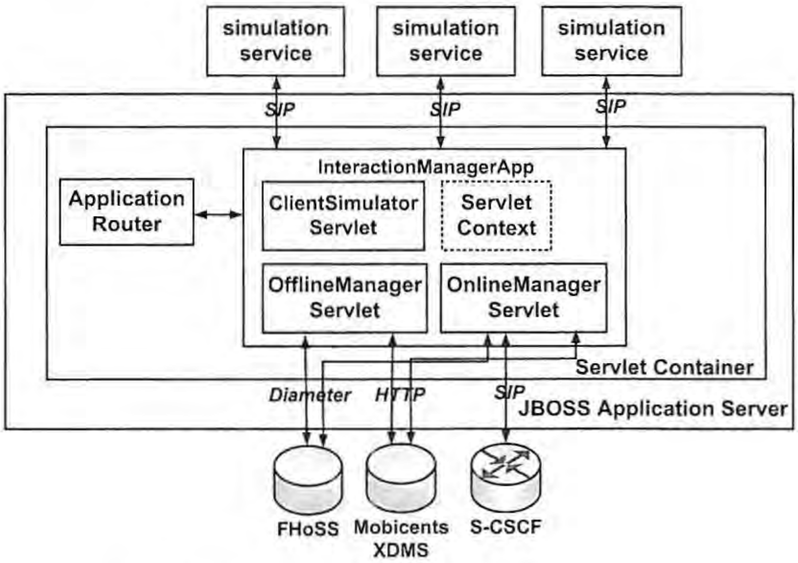


Figure 6.8: SCIM modeled as servlet container.

request to the service node and awaits the response. It performs this operation for all service nodes until all services have been executed. This interaction can be modeled using the servlet container and some servlet applications. Figure 6.8 provides a figure that shows how this was done. The offline and online interaction management functions are provided by the OfflineManagerServlet and OnlineManagerServlet applications. InteractionManagerApp is the name of the application that consists of the two servlets.

In this experiment, the IMS subscriber Bob makes a request to the IMS network to communicate with the user Alice. The experiment used the personalised simservs document provided in Listing 6.4 for Bob and Listing 6.5 for Alice as examples. The simservs document for the user Bob contains the OIR and OCB services and the simservs document for the user Alice contains the OIP and ICB services. These documents simulate a multi-component telephony service for IMS users. In these documents, the priorities of the simulation services are provided to indicate operator policies for these services. The priorities are specified through an attribute named priority. It is possible to incorporate such extensions into the simservs document since the schema provides a

$$< xs : anyAttribute\, namespace = "\#\#any"\; processContents = "lax" / >$$

portion that allows new attributes to be imported from any namespace even outside the simservs schema.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <simservs
  xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
  xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <originating-identity-presentation-restriction active="true" priority="0">
    <default-behaviour>presentation-not-restricted</default-behaviour>
  </originating-identity-presentation-restriction>

  <outgoing-communication-barring active="true" priority="1">
    <cp:ruleset>
      <cp:rule id="rule66">
        <cp:conditions>
          <cp:identity>
            <cp:one id="sip:mallory@open-ims.test"/>
          </cp:identity>
        </cp:conditions>
        <cp:actions>
          <cp:allow>false</cp:allow>
        </cp:actions>
      </cp:rule>
    </cp:ruleset>
  </outgoing-communication-barring>

</simservs>
```

Listing 6.4: Simulation services document for the IMS user Bob.

```
<?xml version="1.0" encoding="UTF-8"?>
    <simservs
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy">

    <incoming-communication-barring active="true" priority="0">
        <cp:ruleset>
            <cp:rule id="rule55">
                <cp:conditions>
                    <cp:identity>
                        <cp:one id="sip:bob@open-ims.test"/>
                    </cp:identity>
                </cp:conditions>
                <cp:actions>
                    <cp:allow>false</cp:allow>
                </cp:actions>
            </cp:rule>
        </cp:ruleset>
    </incoming-communication-barring>

</simservs>
```

Listing 6.5: Simulation services document for the IMS user Alice.

6.8.2.1 Modeling the Simulation Services

The emphasis of the experiment was to demonstrate the composition model of the servlet container. As such, only stub implementations of the simulation services were provided. Four instances of the Kamailio server were used, one for each simulation service involved for the two users.

6.8.2.2 Originating Identity Service

For the originating identity services, there are several interactions that would need to be modeled to provide the full service. Typically, a UE generates a SIP INVITE request and the IMS Core routes the request to the application server that is responsible for handling it. When the OIR service receives the request, it tries to establish if the user has a subscription to the OIR service. If so, it appends a Privacy header to it and passes it on. When the OIP service receives the request, it tries to establish if the terminating user has a subscription to the OIP service. If not, it removes the Privacy header if it finds one in the request and anonymises any headers that provide any identity information that exposes the originating user. If the terminating user has a subscription, the OIP service appends the P-Asserted-Identity header to the SIP message and removes any

Privacy header it finds in the message. To model these behaviours, two instances of Kamailio were used, one for the OIP service and another for the OIR service. The OIP and OIR services simply proxy the request back to the servlet container.

6.8.2.3 Call Barring Service

For the communication barring services, the OCB service rejects the request if the properties of the request violate the rules governing the outgoing service. It does so by sending a SIP 603 (Decline) response to the originating party. Similarly, the ICB service rejects the request if it violates the rules governing the incoming service. To model this behaviour, two Kamailio instances were used, one for the OCB service and another for the ICB service. According to the preferences indicated in the simservs documents, the OCB service simply proxies the request back to the container and the ICB service rejects the call with a SIP 603 (Decline) response.

The DAR configuration file that was used for the experiment is given below.

```
INVITE: ("org.mobicents.servlet.sip.InteractionManagerApp",
"DAR:From", "ORIGINATING", "", "NO_ROUTE", "0","")
REGISTER=("org.mobicents.servlet.sip.InteractionManagerApp",
"DAR\:From", "TERMINATING", "", "NO_ROUTE", "0")
```

Since the two servlets belong to the same application, `InteractionManagerApp` appears twice in the DAR. INVITE requests are handled by the `OnlineManagerServlet` and REGISTER requests are handled by the `OfflineManagerServlet`.

6.8.2.4 Services Address List

Since the simservs document does not contain the network addresses of the simulation services, the SCIM must be able to retrieve this information from elsewhere. For the experiment, it was determined that it would be sufficient for the network addresses to be hard-coded into the Configuration Repository. Each context parameter element consisted of a service name as the parameter name and the IP address and port number made up the parameter value. It is expected that in a real world scenario, the SCIM would either retrieve this information from a network database or be provided with it by an administrator. Ultimately, the SCIM will need to store this information locally. In which case, the Configuration Repository would be an ideal place for it.

6.8.2.5 Simulation Service Invocation

In the experiment, the `OnlineServlet` application represented the core SCIM module for online interaction management. To manage the SIP interactions and sessions, the servlet uses a B2BUA module provided to it by the servlet API. A `doInvite` event handler was provided to it so it could receive SIP INVITE requests. When the servlet receives a request it extracts the user identity from the From header in the SIP INVITE. It then checks to see if the service invocation vector for this user has been created and stored in the application context. If the retrieval attempt returns no results, the servlet downloads user service information from the XDMS and generates the vector at runtime.

Once the vector has been obtained, the servlet examines it to determine which simulation service to invoke first. It does so by identifying the service with the highest priority. For Bob, this is the OCB service. The servlet extracts the network address of the OCB service and sends a request to the indicated location. According to the policies, Bob should not be allowed to make outgoing calls to Mallory, but is allowed to make any other calls. To mimic the OCB service evaluating the request against the policy and approving the communication, the Kamailio instance proxies the request back to the servlet container. The next service is the OIR service which has no policies associated with it. The request was dispatched to this Kamailio instance, which simply returned the request back to the container.

Lastly, the servlet dispatches the request to the ICB service. Policies associated with this service are such that the user Bob is barred from communicating with the user Alice. As such, to mimic this situation, the Kamailio instance returns a SIP 603 (Decline) response. The servlet, using a `doErrorResponse` event handler receives this response and forwards it back to Bob. Bob is notified of the barring and the communication session terminates. Figure 6.9 shows the call flow for this experiment.

6.9 Limitations of the Implementation

The Mobicents SIP Servlets container was a suitable choice to model the SCIM from the perspective of providing a multi-protocol agent that is able to manipulate the SIP protocol and communicate with the XDMS via XCAP. The SCIM in EISL is required to examine policies that apply to service capabilities and to use these policies to determine the invocation of services. In the experiment, only support for establishing the invocation order was tested and the Composition Rules Cache did not contain the policies themselves. In this way, the policy evaluation

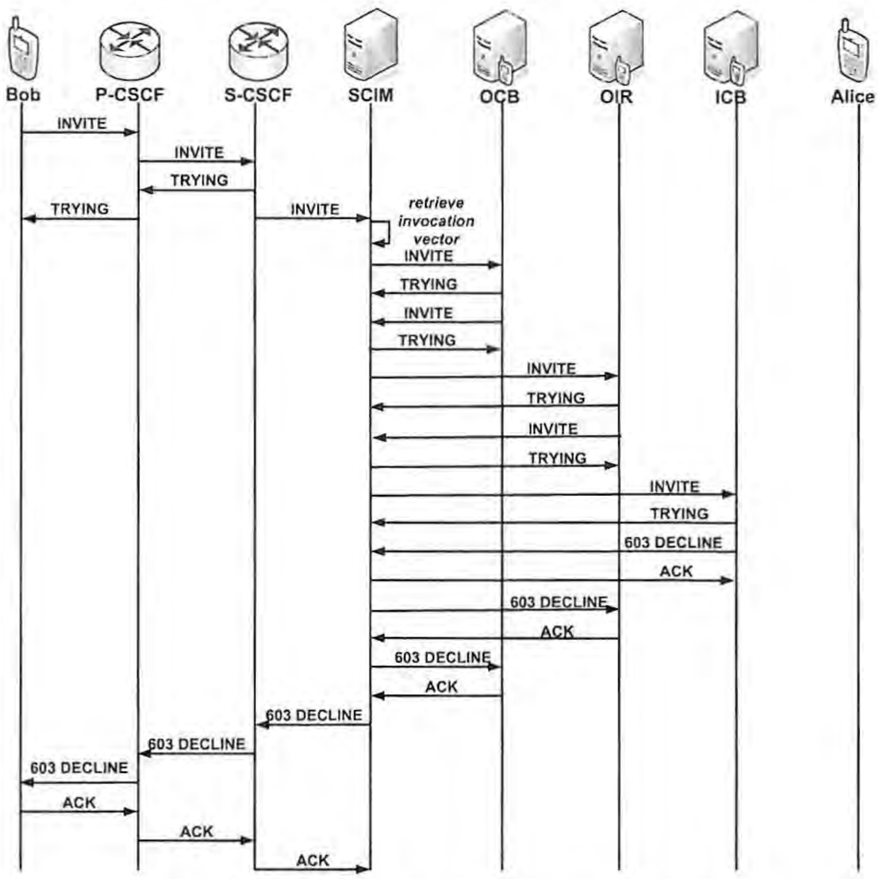


Figure 6.9: Call flow for the complex telephony service.

capabilities of the SCIM were not demonstrated in this experiment, but from previous experiments on service personalisation, it is evident that the SCIM can query the XDMS for portions of the file and determine policies that must be applied. These could then be inserted into the the service invocation vector as additional parameters.

The structure of the simulation services document discussed in this chapter does not take into consideration that each simulation service can belong to more than one service. The documents that were used in the experiment assume that the simulation services constitute a single, prioritised, multi-component service. Thus the implementation does not demonstrate the re-usability property of service capabilities, only the composition of services from many parts.

The implementation also does not demonstrate the feature interaction properties of the EISL design. Part of this requirement was provided by the historical vector which would be used in practice to monitor feature interactions. This SCIM feature is left as future work.

6.10 Discussion

This chapter has described the use of the Mobicents XDMS, the Mobicents SIP Servlet container and a set of SIP servlet applications to provide a prototypical testbed that models the extended IMS layer that has been proposed. The choice of using SIP servlets over JAIN SLEE was motivated by the much easier programming model and the comparable invocation mechanism which is supported by the default application router. The Diameter and XCAP client libraries that the project provides also make it easy to implement the required protocol interfaces that belong to the SCIM. The Mobicents XDMS provides a simple framework for the incorporation of new appusages not currently defined, which made it a suitable candidate for modeling the service repositories. The simservs appusage provided an effective way of supporting both types of service repositories by using the global simservs document as the service repository and the user simservs documents as personalisable user repositories. The use of common policy features and the extended rule conditions defined by the appusage provide a powerful platform for service personalisation. The ability to interleave simulation services using the default application router and policy rules fulfills the dynamic requirements of online interaction management and therefore provides a suitable basis for the development of the SCIM component of the IMS service broker. The prototype is limited in that it does not provide the policy evaluation and feature interaction management capabilities of the SCIM, but it does fulfill the main objective of providing a proof of concept implementation that tests the validity of the design. The next chapter concludes this thesis.

Chapter 7

Summary and Future Work

This thesis has investigated service capability interaction management in the IMS. It has summarised the literature that has been produced on this topic by organisations such as the 3GPP, and furthermore, has analysed the main arguments that have been presented on this topic by the IMS research community. A novel design named EISL, which stands for the Extended IMS Service Layer, was presented that addresses both the challenges and the opportunities that result from the introduction of a SCIM into the IMS service layer. An EISL implementation that incorporates a SCIM was developed that verifies the suitability of the design to enable service interaction management and offers service personalisation to IMS users. This chapter presents the main conclusions of the thesis, outlining the main contributions of this work and ends by listing some items for future work.

7.1 Summary

This section identifies the main points that are addressed in this thesis and identifies the contributions it has made in those areas.

7.1.1 Framework for Service Creation from Service Capabilities

Technical specifications have been developed for services such as presence, messaging and conferencing that describe how these services are structured and behave in an IMS network. In these specifications, the services have been described in their totality and not in terms of the underlying building blocks that should be used to develop them. Yet standards development organisations such as the 3GPP and ETSI maintain that the most appropriate strategy of service creation is through the use of service capabilities in order to support service interworking and the re-usability of features. The mailing list discussion of the SA1 working group of the 3GPP that was summarised in Section 2.2.5 highlighted some of the challenges related with this strategy. In it, members lamented at the lack of a clear directive on how operators would use service capabilities to deploy certain services. This thesis finds that the inability on the part of the standards development organisations to clearly articulate the process of service creation from service capabilities and to provide a clear framework that could be used by network operators and service providers has hampered, and will continue to hamper the movement towards the composition of services from service building blocks. In addition to hampering service compositions, it has also retarded the introduction of the SCIM into the IMS network since operators are more likely to deploy monolithic services that are self-contained and do not employ the mechanisms that are discussed in this thesis.

7.1.2 SCIM Standardisation

The review of the technical specifications that have been produced by the 3GPP revealed that there is very little that has been decided by them on the architecture of the SCIM. Firstly, the main specification on network architecture declares that the SCIM is an optional node. Secondly, though the 3GPP addresses the potential architectural placement of the SCIM as well as potential algorithms for interaction management, the work item never resulted in concrete decisions being made and incorporated into the main technical specifications on IMS. The lack of momentum in the 3GPP on the issue of the SCIM has led to a large body of work being published by researchers on this topic. However, not all of these are in harmony with the underlying philosophy of extending the IMS, which is based on the re-use of existing protocols whenever and wherever possible, and limiting the impact on other network elements. This thesis has identified the divide that exits between the incomplete technical specifications and the products of the research community. It has provided a bridge between the two and has identified the proposals and procedures that most appropriately address the requirements and are in synergy with the existing architecture.

7.1.3 Extending the IMS Service Layer

The challenge of interaction management in IMS requires the specification of the structure of the SCIM and the identification of entities with which it must interact with, including the protocol interfaces it needs in order to do so. To address the open question regarding the structural architecture of the SCIM, this thesis defines the use of three protocol stacks and three logical components. The protocols used are SIP, Diameter and XCAP which allow it to communicate with the SIP application servers that host service capabilities, the HSS which hosts user profile information and the service repositories that host service information, respectively. A Configuration Repository in the SCIM stores network configuration information such as the location of the service repositories, while the Composition Engine and Composition Rules Cache create and store information that assists the SCIM in handling user requests during the online interaction management stage. The thesis also promotes the use of external repositories that store information about network services. Some of the service information must apply in a global context, while other information applies only to specific users. The XDMS, a standard entity in the IMS service layer in presence-related services, is elected to provide these features. This decision is consistent with the goal of re-using existing IMS protocols and also simplifies the design by using the same technology to provide both types of repositories. The overall impact of the introduction of the SCIM is kept to a minimum since the SCIM uses standard IMS interfaces and does not require a change in the way that standard entities such as the S-CSCF behave.

7.1.4 Software Solutions for Service Development

The investigation that was related in Chapter 5, which explored the practical side of service development and deployment for an IMS testbed, showed that there is a shortage of free and open source application and media service platforms that conform to the standard requirements. While many of the SIP application servers that were reviewed are able to exhibit the required behaviours in terms of manipulating the SIP protocol, most of them fail to provide the required Sh Diameter interface. Of the systems that were reviewed, only Sailfin, Mobicents SIP Servlets and Mobicents JAIN SLEE provide the required Diameter support. Similarly, most of the media service platforms that were reviewed fail to exhibit the required architectural and functional features of the IMS MRF. Protocol implementations of the MEGACO / H.248 interface between the MRFC and the MRFP are scarce, which automatically rules out some of them, and many do not decouple the media control functions from the media processing functions. The Mobicents Media Server was found to be the only system among those that were reviewed that fulfills the requirements. This thesis has exposed the lack of support for IMS compliant service and media platforms on the open Internet. The lack of open systems with which to experiment is likely to hamper the efforts of the average developer to gain experience in service development and deployment for IMS. In turn, this could have a negative impact on the popularity of the IMS as a docking station for services. However, this thesis has found that the Mobicents communication platform qualifies it above many other application and media platforms as an important tool for testbed engineers, and can greatly assist developers in learning more about the complex IMS platform and the dynamics of its service integration support.

7.1.5 Technical Feasibility of Implementing the Design

One of the properties of a good design is independence from implementational biases. Though this is true, in the case of extending IMS, forcing the re-use of existing IMS protocols is an important qualifier to this goal. EISL satisfies these conditions as it uses standard IMS protocols to support its operations and all protocol interfaces are used in their existing state without any extensions. In addition, as far as the protocols themselves are concerned, EISL does not require any specific proprietary stack to be used, only that those stacks adhere to the standards. Ultimately, none of the IMS Core entities are modified in the design, which has helped eliminate any negative impact that the introduction of the SCIM has on the IMS service layer.

7.2 Future Work

7.2.1 Considerations for Architecture Alternatives

Though the discussion on the EISL design was done from the perspective of a centralised placement of the SCIM, efforts were made to describe how the mechanisms that have been employed would apply in a distributed scenario. The design should benefit from decentralisation, given the general consensus in the computing world on the benefits of decentralisation, such as increased availability and fault-tolerance. Theoretically, EISL should support decentralisation. For the S-CSCF to communicate with multiple SCIMs, all that is required is for the HSS to index different SCIMs in the iFC portions of the user profiles belonging to different IMS subscribers. The same applies for offline interaction management, where the SCIM would have third party registration requests configured into the user profiles of the subscribers they serve. The SCIMs would need to have their Diameter stacks configured to point to the HSS, just like all SIP application servers currently do. In so doing, the network uses existing mechanisms to partition the user base among multiple SCIMs. For future work, it would be beneficial to prove these claims by means of an implementation. Such work would be able to shed light on the benefit versus cost of such interventions. In addition, the design does not cover the role of peered relationships between SCIMs. If application servers are bound to specific SCIMs, then a request from a user to a SCIM for a service for which it is not responsible, might be best handled by issuing a request to a peer SCIM for assistance in handling the request. Additional mechanisms would be needed to handle the complete set of interactions between a SCIM and the application servers and a SCIM and another SCIM.

7.2.2 Removal of Service Subscriptions and Notification of new Services

The global service repository contains all the services that the network offers to its users, but by definition, an administrator has the ability to update the global document when there is a need. One of the changes that can be made is the removal of a service which is no longer being offered by the network operator or service provider. In this case, it is necessary to define resource dependencies that will assert integrity checks to ensure that the corresponding service is removed from the respective user home directories if it is no longer being offered. The Mobicents XDM allows developers to define authorisation objects that extend the `AuthorizationPolicy` class to support such functionality. For example, the `RLSServicesAuthorizationPolicy` class has been developed to ensure that there is data consistency between the user resource lists and the rls-services documents that reside on the server. For future work, a possible extension to the design would be to provide a class that extends the authorisation policy class in Mobicents to ensure that when a service is removed from the global document, the same information is removed from the user directories.

The other possibility is that the network operator or service provider adds a new service to the service repository. The current design would be able to reflect this addition to the repository to users who join the network after the fact or to those users who have not personalised their services yet. However, it would not be able to provide this information to users who have already personalised their services and have no need to interrogate the global simservs document. These users would be deprived access to any newly added services. For future work, a possible extension would be to augment the design with presence abilities so that a watcher could issue subscription requests to the global document in order to be notified of changes to it. Due to the sensitivity of the global document, it may be appropriate to setup the SCIM to be the only watcher with privileges to issue subscription requests for such updates. It would be the responsibility of the SCIM to effect changes in the user documents of the users it is serving. This solution would also be suitable for solving the problem of deleted services. In both cases, the network could send the user an instant message or play an announcement to alert the user that a new service is available, or that a subscribed service has been removed.

7.2.3 Custom Application Router for Composed Services

The DAR in the Mobicents servlet container is a bare bones application router. By definition, an application router should be able to interrogate external repositories in order to make an informed decision about which servlet application to invoke next. Rather, this function was supplied by the servlet applications themselves which needed to interrogate the service repository in order to gain access to the user service bindings and policy conditions. For future work, to extend the implementation provided in this thesis, it would be beneficial if a customised version of the application router were developed that would be able to take information in the service repository into consideration when invoking applications so as to appropriately invoke servlet applications based on policy information.

7.2.4 Evolution Towards Service Brokering

In the opening chapters, the thesis described the confusion in the research community regarding the terms SCIM and service broker. However, it quickly established the difference between the two, providing a description of the functions of each and showed that the SCIM should be regarded as a component of the service broker. Section 3.6 provided a list of the components of the service broker. The list shows that the service broker has three policy components, namely the policy evaluator, the policy enforcer and the policy manager. The common policy features of the service repositories partially address the policy requirements of the service broker by providing facilities for the management of user and operator policies for service execution through the use of XCAP. The SCIM evaluates and enforces these policies when it receives a message from a user. However, policies are not only applied to realtime requests by IMS subscribers, but must also address the control of service requests from third party service providers. This represents a different use of policies, as these policies would have to permit or prohibit discovery and execution of services by applications and not necessarily by users. For future work, the design of EISL would need to incorporate this class of policies into the service capabilities listed in the service repository and would also need to provide mechanisms for exposing service capabilities to third parties.

7.3 Conclusion

This thesis has presented a proposal for the structural and functional architecture of the SCIM — an IMS node which has not yet been standardised. It has shown that the current IMS service layer must be extended in order to fully support the role of an interaction manager and that existing IMS technologies are sufficient for this. A SCIM can be imported into the service layer without impacting greatly on the structure of the current service layer. It is unclear when the SCIM will be standardised but when it does it will provide important cost savings to the service providers and operators and will enable greater user involvement in the execution of services.

References

[1] ETSI. Organisation Chart. Available Online, August 2010. URL: http://www.etsi.org/WebSite/AboutETSI/EtsiOrganizationchart.aspx.

[2] H Holma and A Toskala. *LTE for UMTS: OFDMA and SC-FDMA based Radio Access*. John Wiley and Sons, Ltd, First edition, 2009.

[3] M Poikselka and G Mayer. *The IMS IP Multimedia Concepts and Services*. Wiley, Third edition, 2009.

[4] 3GPP. TS 22.105: Services and Service Capabilities. Third Generation Partnership Project, December 2008.

[5] 3GPP. TS 22.101: Service Aspects; Service Principles. Third Generation Partnership Project, December 2008.

[6] 3GPP. TS 22.127: Service Requirement for OSA. Third Generation Partnership Project, March 2008.

[7] ETSI. TR 181.004: NGN Generic Capabilities and their use to Develop Services. European Telecommunication Standards Institute, March 2006.

[8] 3GPP. TS 23.218: IP Multimedia (IM) Session Handling; IM Call model; Stage 2. Third Generation Partnership Project, December 2008.

[9] G Camarillo and M Garcia-Martin. *The 3G IP Multimedia Subsystem (IMS)*. John Wiley and Sons, Ltd, Third edition, 2008.

[10] Q Qi, J Liao, X Zhu, and Y Cao. DSCIM: A Novel Service Invocation Mechanism in IMS. In *GLOBECOM '08: IEEE Global Telecommunications Conference*, pages 1 –5. IEEE Computer Society, November 2008.

[11] OMA. OMA Service Environment Version 1.0.5. Open Mobile Alliance, October 2009.

[12] A Gouya and N Crespi. Service Broker for Managing Feature Interactions in IP Multimedia Subsystem. In *ICN '07: Sixth International Conference on Networking*, pages 54–59, April 2007.

[13] 3GPP. TS 24.623: Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulation Supplementaty Services. Third Generation Partnership Project, December 2009.

[14] 3GPP. 3GPP Releases. Available Online, August 2010. URL: http://www.3gpp.org/releases.

[15] 3GPP. TS 29.329: Sh Interface based on the Diameter Protocol; Protocol details. Third Generation Partnership Project, March 2009.

[16] M Tsietsi, A Terzoli, and G Wells. Mobicents as a Service Deployment Environment for OpenIMSCore. In *SATNAC '09: 12th Southern African Telecommunications Networks and Applications Conference*, September 2009.

[17] Miniwatts Marketing Groups. Internet World Stats. Available Online, January 2011. URL: http://www.internetworldstats.com/stats.htm.

[18] P Newman. In Search of the All-IP Mobile Network. *Communications Magazine, IEEE*, 42(12):S3 – S8, December 2004.

[19] Y Lin, A Pang, Y Haung, and I Chlamtac. An All-IP Approach for UMTS Third-Generation Mobile Networks. *Network, IEEE*, 16(5):8 – 19, September 2002.

[20] ITU. About ITU. Available Online, August 2010. http://www.itu.int/net/about/index.aspx.

[21] C Shumin. Current Development of IMT-2000 in China. *Communications Magazine, IEEE*, 36(9):157 –159, September 1998.

[22] ITU. Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000. Recommendation M.1225. International Telecommunication Union, February 1997.

[23] ETSI. About ETSI. Available Online, August 2010. URL: http://www.etsi.org/WebSite/AboutETSI/AboutEtsi.aspx.

[24] 3GPP. About 3GPP. Available Online, October 2009. URL: http://www.3gpp.org/About-3GPP.

[25] 3GPP2. About 3GPP2. Available Online, October 2009. URL: http://www.3gpp2.org/Public_html/Misc/AboutHome.cfm.

[26] IETF. About the IETF. Available Online, May 2009. URL: http://www.ietf.org/about/.

[27] K Rosenbrock, R Sanmugam, S Bradner, and J Klensin. RFC 3113: 3GPP-IETF Standardization Collaboration. Internet Engineering Task Force, June 2001.

[28] S Bradner, P Calhoun, H Cuschieri, S Dennett, G Flynn, M Lipford, and M McPheters. RFC 3131: 3GPP2-IETF Standardization Collaboration. Internet Engineering Task Force, June 2001.

[29] M Garcia-Martin. RFC 4083: Input 3rd-Generation Partnership Project (3GPP) Release 5 Requirements on the Session Initiation Protocol (SIP). Internet Engineering Task Force, May 2005.

[30] OMA. About OMA. Open Mobile Alliance, January 2010. URL: http://www.openmobilealliance.org/AboutOMA/Default.aspx.

[31] OMA. Utilization of IMS Capabilities Architecture Version 1.0. Open Mobile Alliance, August 2005.

[32] OMA. Enabler Release Definition for IMS in OMA Version 1.0. Open Mobile Alliance, August 2005.

[33] H Holma and A Toskala. *WCDMA for UMTS*. John Wiley and Sons, Ltd, Fifth edition, 2010.

[34] ITU. ITU-T Recommendation I.130: Method for the Characterization of Telecommunication Services Supported by an ISDN and Network Capabilities of an ISDN. International Telecommuication Union (ITU), 1988.

[35] 3GPP. TS 23.198: Open Service Access; Stage 2. Third Generation Partnership Project, December 2006.

[36] P Calhoun, J Loughney, E Guttman, G Zorn, and J Arkko. RFC 3558: Diameter Base Protocol. Internet Engineering Task Force, September 2003.

[37] M Tsietsi, A Terzoli, and G Wells. Towards Service Capability Interaction Management in IMS Networks. In *SATNAC '10: 13th Southern African Telecommunications Networks and Applications Conference*, September 2010.

[38] 3GPP. 3GPP - SA 1 - Services. Available Online, September 2010. http://www.3gpp.org/SA1-Services.

[39] 3GPP. TR 21.900: Technical Specification Group Services and System Aspects; Technical Specification Group working methods. Third Generation Partnership Project, December 2009.

[40] 3GPP. TS 22.004: General on Supplementary Services. Third Generation Partnership Project, December 2009.

[41] 3GPP. TS 22.228: Service Requirements for the Internet Protocol IP Multimedia Core Network Subsystem; Stage 1. Third Generation Partnership Project, December 2008.

[42] J Rosenberg. RFC 4825: The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). Internet Engineering Task Force, May 2007.

[43] 3GPP. TS 23.228: IP Multimedia Subsystem; Stage 2. Third Generation Partnership Project, September 2009.

[44] J Rosenburg, H Schulzrinne, G Camarillo, A Johnston, J Peterson, R Sparks, M Handley, and E Schooler. RFC 3261: SIP: Session Initiation Protocol. Internet Engineering Task Force, June 2002.

[45] 3GPP. TS 23.008: Organization of Subscriber Data. Third Generation Partnership Project, March 2010.

[46] 3GPP. TS 23.002: Network Architecture. Third Generation Partnership Project, December 2008.

[47] H Khlifi and J C Gregoire. IMS Application Servers: Roles, Requirements, and Implementation Technologies. *Internet Computing, IEEE*, 12(3):40 –51, May 2008.

[48] A Cuevas, JI Moreno, P Vidales, and H Einsiedler. The IMS Service Platform: A Solution for Next-Generation Network Operators to be More than Bit Pipes. *Communications Magazine, IEEE*, 44(8):75 –81, 2006.

[49] 3GPP. TS 24.141: Presence Service using the IP Multimedia (IM) Core Network (CN) subsystem. Third Generation Partnership Project, December 2008.

[50] 3GPP. TS 24.247: Messaging Service using the IP Multimedia (IM) Core Network (CN) subsystem. Third Generation Partnership Project, December 2008.

[51] 3GPP. TS 24.147: Conferencing Service using the IP Multimedia (IM) Core Network (CN) subsystem. Third Generation Partnership Project, December 2008.

[52] A Gouya, N Crespi, E Bertin, and L Oueslati. Managing Service Capability and Service Feature Interactions in the IMS of UMTS. In *Proceedings of the International Conference on Networking and Services*, pages 50–55, Washington, DC, USA, 2006. IEEE Computer Society.

[53] Y Wang, K Yangi, R Li, and L Zhang. An Improved SCIM-based Service Invocation Mechanism for Integrated Services in IMS. In *Mobility '08: Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, pages 1–5, New York, NY, USA, 2008. ACM.

[54] 3GPP. AW: Tdoc S1-010232 - Standardisation of IMS services; Comments. Available Online. URL: http://list.etsi.org/scripts/wa.exe?A2=ind0102&L=3gpp_tsg_sa_wg1&T=0&P=4522.

[55] 3GPP. Study on Architecture Impacts of Service Brokering - 3GPP Specification Detail. Available Online, October 2010. URL:http://www.3gpp.org/ftp/Specs/html-info/23810.htm.

[56] 3GPP. TR 23.810: Study on Architecture Impacts of Service Brokering. Third Generation Partnership Project, September 2008.

[57] R Spiers and N Ventura. Dynamic Service Orchestration in the IP Multimedia Subsystem. In *SATNAC '09: Southern African Telecommunications Networks and Applications Conference*, September 2009.

[58] K Kocan, W Roome, and V Anupam. Service Capability Interaction Management in IMS Using the Lucent Service Broker Product. *Bell Labs Technical Journal*, 10(4):1–16, 2006.

[59] N Xia and W Zhai. Study on IMS Service Broker. In *ICCIT '08: Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology*, pages 340–343, Washington, DC, USA, 2008. IEEE Computer Society.

[60] A Gouya, N Crespi, and E Bertin. SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture. In *ICC '06: IEEE International Conference on Communications*, pages 1748–1753. IEEE, June 2006.

[61] X Wu, J Buford, K Dhara, V Krishnaswamy, and M Kolberg. Feature interactions between internet services and telecommunication services. In *IPTComm '09: Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, pages 10:1–10:12, New York, NY, USA, 2009. ACM.

[62] H Liu and F Yang. A Generic Approach to Service Conflict Control in IMS. In *Networking and Services, 2009. ICNS '09. Fifth International Conference on*, pages 444 –449, 2009.

[63] N Crespi. A Distributed Mechanism to Resolve Dynamically Feature Interaction in the UMTS IP Multimedia Subsystem. In *ASWN '06: Sixth International Workshop on Applications and Services in Wireless Networks (ASWN)*, pages 199–206, May 2006.

[64] T Magendanz, K Knuttel, and D Witaszek. Service Delivery Platform Options for Next Generation Networks within the National German 3G Beyond testbed. In *SATNAC '04: Proceedings of the 7th South African Telecommunications Networks and Applications Conference*, September 2004.

[65] N. Blum, I. Boldea, T. Magedanz, U. Staiger, and H. Stein. A Service Broker Providing Real-Time Telecommunications Services for 3rd Party Services. In *COMPSAC '09: Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference*, pages 85–91, Washington, DC, USA, 2009. IEEE Computer Society.

[66] OMA. Policy Evaluation, Enforcement and Management Architecture Version 1.0. Open Mobile Alliance, August 2008.

[67] M Pitman and N Ventura. Aggregation Efficacy of Resource List Servers in IMS Presence Services. In *Proceedings of the Southern African Telecommunications Networks and Applications Conference*, Wild Coast, South Africa, 2008.

[68] Fraunhofer FOKUS NGNI. The Open Source IMS Core Project. Available Online, June 2010. URL: http://www.openimscore.org/.

[69] Fokus Fraunhofer. Next Generation Network Infrastructures. Available Online, June 2010. URL: http://www.fokus.fraunhofer.de/en/ngni/index.html.

[70] iptel.org. About iptel.org. Available Online, December 2009. URL: http://www.iptel.org/about.

[71] Fokus Fraunhofer. The Open IMS Core CSCFs. Available Online, December 2009. URL: http://www.openimscore.org/docs/ser_ims/index.html.

[72] FOKUS Fraunhofer. Overview of the HSS. Available Online, November 2010. URL: http://www.openimscore.org/docs/FHoSS/index.html.

[73] C Bauer and G King. *Understanding Object/Relational Persistence*. Manning Publications, Second edition, 2007.

[74] M. Tsietsi, Z. Shibeshi, A. Terzoli, and G. Wells. An Asterisk-Based Framework for e-Learning using Open Protocols and Open Source Software. In *Innovations for Digital Inclusions, 2009. K-IDI 2009. ITU-T Kaleidoscope:*, pages 1 –5, 312009-sept.1 2009.

[75] Digium. Asterisk- Open Source Telephony Projects. Available Online, November 2010. URL:http://www.asterisk.org.

[76] J van Meggelen, J Smith, and L Madsen. *Asterisk: The Future of Telephony*. O'Reilly Media, First edition, 2005.

[77] Digium. Diameter Implementation of Asterisk CDR. Available Online, December 2010. URL: https://issues.asterisk.org/view.php?id=15006.

[78] Kamailio. Kamailio SIP Server. Available Online, December 2010. URL: http://www.kamailio.org.

[79] SIP Router. The SIP Router Project. Available Online, December 2010. URL:http://sip-router.org.

[80] A K MacDonald. Configuration and Deployment of an IMS Testbed. Technical report, Laboratorio de Redes Avanzadad, Instituto Tecnologico Autonomo de Mexico, 2009.

[81] AG Projects. OpenXCAP - Free XCAP Server for SIP SIMPLE (RFC 4825). Available Online, April 2009. URL: http://openxcap.org/.

[82] Java Community Process. JSR 116: SIP Servlet API. Available Online, October 2001. URL: http://jcp.org/en/jsr/detail?id=116.

[83] Java Community Process. JSR 289: SIP Servlet v1.1. Available Online, May 2009. URL: http://jcp.org/en/jsr/detail?id=289.

[84] A Kristen. SIP Servlet API 1.0. Available Online, Febriuary 2003. URL: http://www.wesip.com/mediawiki/API/sipservlet-1.0-fcs.pdf.

[85] Oracle. Sailfin Project - Project Kenai. Available Online, February 2011. http://sailfin.java.net/.

[86] Oracle. Overview - Sun Diameter API JavaDoc. Available Online, February 2011. URL: http://sailfin.java.net/nonav/diameter/apidocs/index.html.

[87] JBoss. SIP Servlets Server User Guide. Available Online, February 2011. URL: http://hudson.jboss.org/hudson/view/Mobicents/job/MobicentsBooks/lastSuccessfulBuild /artifact/sip-servlets/index.html.

[88] JBoss. Mobicents Diameter User Guide. Available Online, February 2011. URL: http://hudson.jboss.org/hudson/job/MobicentsBooks/lastSuccessfulBuild/artifact/diameter /core/index.html.

[89] H Hakala, L Mattila, J-P Koskinen, M Stura, and J Loughney. RFC 4006: Diameter Credit-Control Application. Internet Engineering Task Force, August 2005.

[90] T Mecklin, M Opsenica, H M Rissanen, and D Valderas. ImsInnovation - Experiences of an IMS testbed. *Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on*, 0:1–6, 2009.

[91] Java Community Process. JSR 240: JAIN SLEE (JSLEE) v1.1. Available Online, October 2008. URL: http://jcp.org/en/jsr/detail?id=240.

[92] Java Community Process. JSR 23: JAIN MGCP API Specification. Available Online, May 2009. URL: http://jcp.org/en/jsr/detail?id=23.

[93] Java Community Process. JSR 32: JAIN SIP API Specification. Available Online, May 2009. URL: http://jcp.org/en/jsr/detail?id=32.

[94] M Tsietsi, A Terzoli, and G Wells. An Open Service Delivery Platform for adding value to Softswitch-Based Telephony Environments. In *SATNAC '08: 11th Southern African Telecommunications Networks and Applications Conference*, September 2008. Work in progress.

[95] Red Hat. Mobicents - The Open Source SLEE and SIP Server. Available Online, May 2011. URL: http://www.mobicents.org.

[96] S Tarkoma, J Rovira, E Postmann, H Rajasekaran, and E Kovacs. Creating Converged Services for IMS Using the SPICE Service Platform. In *ICIN '07: International Conference on Intelligence in Service Delivery Networks*, October 2007.

[97] ITU. ITU-T H.248: Gatway Control Protocol: Version 3. International Telecommunication Union, September 2005.

[98] M Arango, A Dugan, I Elliot, C Huitema, and S Pickett. RFC 2705: Media Gateway Control Protocol (MGCP) Version 1.0. Internet Engineering Task Force, October 1999.

[99] K L Larsen, G Castro, H P Schwefel, and V Scotto di Carlo. Corporate Convergence with the 3GPP IP Multimedia Subsystem. *Next Generation Mobile Applications, Services and Technologies, International Conference on*, 0:29–35, 2007.

[100] iptel.org. SIP Express Media Server. Available Online, December 2009. URL: http://www.iptel.org/sems.

[101] JBoss. Mobicents Media Server User Guide. Available Online, February 2011. URL: http://hudson.jboss.org/hudson/job/MobicentsBooks/lastSuccessfulBuild/artifact/media /index.html.

[102] F Cuevo, N Greene, A Rayhan, C Huitema B Rosen, and J Segers. RFC 3015: Megaco Protocol Version 1.0. Internet Engineering Task Force, November 2000.

[103] J Van Dyke, E Burger, and A Spitzer. RFC 4722: Media Server Control Markup Langage (MSCML) and Protocol. Internet Engineering Task Force, November 2006.

[104] Java Community Process. JSR 309: Media Server Control API. Available Online, May 2010. URL: http://jcp.org/en/jsr/detail?id=309.

[105] JBoss. Converged Application Demo. Available Online, February 2011. URL: http://groups.google.com/group/mobicents-public/web/converged-application-demo?hl=en.

[106] 3GPP. IP Multimedia Subsystem (IMS) based Packet Switch Streaming (PSS) and Multimedia Broadcast/Multicast Service (MBMS); User Service; Protocols. Third Generation Partnership Project, December 2009.

[107] InexBee. Mercuro IMS Client. Available Online, February 2011. URL: http://www.mercuro.net/.

[108] FOKUS Fraunhofer Institute. myMONSTER - Telcom Communicator Suite. Available Online, February 2011. URL: http://www.monster-the-client.org/.

[109] PT Inovacao. IMS Communicator. Available Online, February 2011. URL: http://imscommunicator.berlios.de/.

[110] University of Cape Town. UCT IMS Client. Available Online, June 2010. URL: http://uctimsclient.berlios.de.

[111] B Day. Progress on the JMF and Java Media APIs. Available Online, March 2011. URL: http://www.javaworld.com/jw-06-1999/jw-06-media.html?page=2.

[112] Sourceforge.net. Jffmpeg - Java Audio and Video Codecs for JMF. Available Online, February 2011. URL: http://jffmpeg.sourceforge.net/.

[113] Sourceforge.net. FMJ - Freedom for Media in Java. Available Online, February 2011. URL: http://fmj-sf.net/.

[114] D Waiting, R Good, R Spiers, and N Ventura. The UCT IMS Client. In *TRIDENT '08: Third International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, April 2009.

[115] E Burger. RFC 4483: A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages. Internet Engineering Task Force, May 2006.

[116] JBoss. Mobicents RTSP Stack 1.0.0.BETA1 Released. Available Online, February 2011. URL: http://mobicents-media-server.blogspot.com/2009/09/mobicents-rtsp-stack-100bta1-released.html.

[117] R Spiers, R Marston, R Good, and N Ventura. The UCT IMS IPTV Initiative. In *NGMAST '09: Third International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 503–508. IEEE Computer Society, September 2009.

[118] ETSI. TS 182.027: IPTV Architecture; IPTV functions supported by the IMS subsystem. European Telecommunication Standards Institute, July 2009.

[119] M Tsietsi, R Musvibe, A Terzoli, and G Wells. Towards IPTV Service Discovery and Selection in an IMS environment. In *ICUMT '10: International Congess on Ultra Modern Telecommunications and Control Systems and Workshops*, pages 195 –201, 2010.

[120] M Tsietsi, Z Shibeshi, S Ndakunda, and R Musvibe. Mobicents Converged Demo with Video Streaming. Available Online, October 2009. URL: http://www.youtube.com/watch?v=YvzeYvWtDaI.

[121] C Chrighton, D T Long, and D C Page. JAIN SLEE vs SIP Servlet Which is the best choice for an IMS application server? In *Telecommunication Networks and Applications Conference, 2007. ATNAC 2007. Australasian*, pages 448 –453, 2007.

[122] S Bessler, J Zeiss, R Gabner, and J Gross. *An Orchestrated Execution Environment for Hybrid Services*, chapter Teil II - Verteilte Anwendungen und Web Services, pages 77–88. Springer Berlin Hedelberg, 2007.

[123] Ivelin Ivanov. JAIN SLEE, SIP Servlets and Parlay/OSA. Available Online, August 2005. http://ivelinivanov.blogspot.com/2005/08/jain-slee-sip-servlets-and-parlayosa.html.

[124] M Tsietsi, A Terzoli, and G Wells. Using JAIN SLEE as an Interaction and Policy Manager for Enabler-based Services in Next Generation Networks. *International Journal of Electronics and Telecommunciations*, 56(2):117–124, June 2010.

[125] M Tsietsi, A Terzoli, and G Wells. JAIN SLEE as a Service Capability Interaction Manager for Open IMS Core. In *BROADBANDCOM '09: 4th International Conference on Broadband Communication, Information Technology and Biomedical Applications*, July 2009.

[126] 3GPP. TS 24.173: IMS Multimedia Telephony Communication Service and Supplementary services; Stage 3. Third Generation Partnership Project, March 2010.

[127] 3GPP. TS 22.127: Multimedia Telephony Service and Supplementary Services; Stage 1. Third Generation Partnership Project, March 2009.

[128] H Schulzrinne, H Tschofenig, J Morris, J Cuellar, J Polk, and J Rosenberg. RFC 4745: Common Policy: A Document Format for Expressing Privacy Preferences. Internet Engineering Task Force, February 2007.

[129] OMA. XML Document Management (XDM) Specification - Draft version 1.1. Open Mobile Alliance, July 2007.

[130] 3GPP. TS 24.408: PSTN/ISDN Simulation services Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR); Protocol specification. Third Generation Partnership Project, March 2008.

[131] 3GPP. TS 24.407: PSTN/ISDN Simulation services Originating Identification Presentation (OIP) and Originating Identification Restriction (OIR); Protocol specification. Third Generation Partnership Project, December 2009.

[132] 3GPP. TS 24.404: PSTN/ISDN Simulation services: Communication Diversion (CDIV); Protocol specification. Third Generation Partnership Project, June 2009.

[133] 3GPP. TS 24.411: PSTN/ISDN Simulation services: Anonymous Communication Rejection (ACR) and Communication Barring (CB); Protocol specification. Third Generation Partnership Project, July 2008.

Appendix A

Additional Diagrams

This appendix provides additional diagrams that were left out of the main body of the thesis for the sake of brevity. They illustrate the four operational modes of a SIP application server and the different possible architectural placements of the SCIM.

A.1 Operational Modes of a SIP Application Server

The SIP application server can behave in four distinct modes. These are as a terminating UA, a proxy, a back-to-back UA and a third party call controller. Application servers can be developed in such a way as to combine these modes in any way and can take on more than one mode of operation during a single multimedia session.

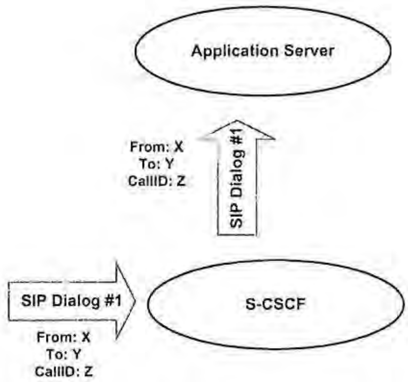


Figure 7.1: SIP application server as a terminating UA, or redirect server.

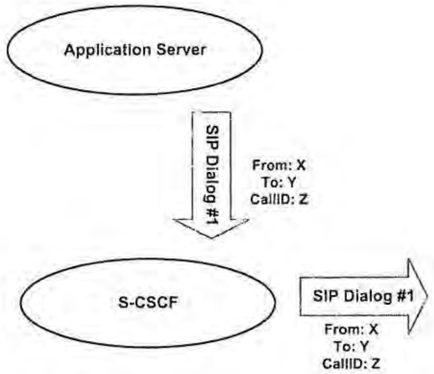


Figure 7.2: SIP application server as an originating UA.

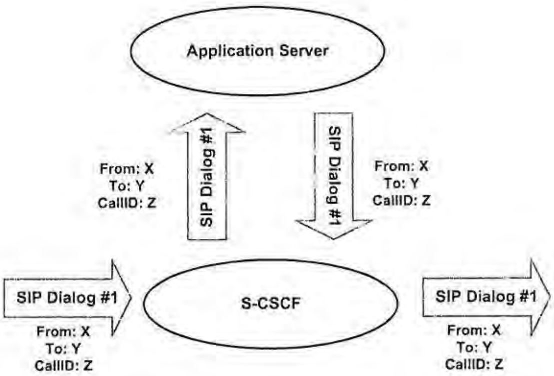
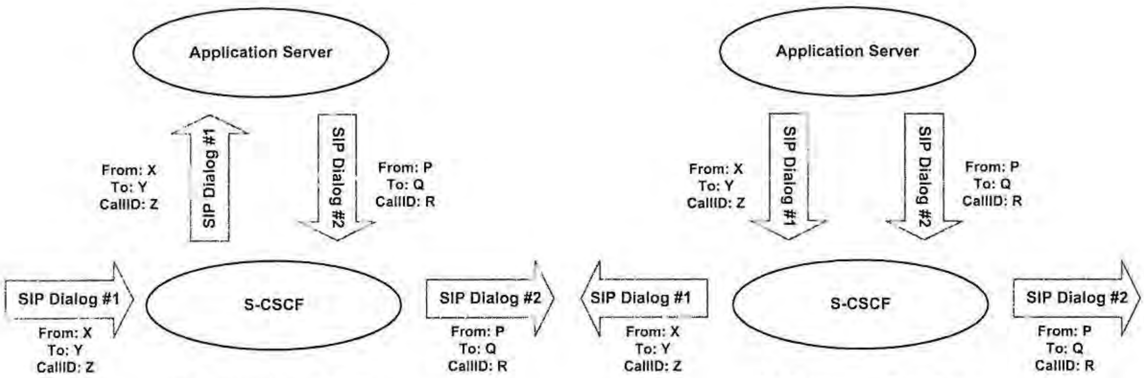


Figure 7.3: SIP application server as a SIP Proxy.



(a) Routeing B2BUA. (b) Initiating B2BUA.

Figure 7.4: SIP application server as a third party call controller (3PCC) with defined B2BUA modes.

A.2 Architectural Placement of the SCIM

In an attempt to address possible architectural placements of the SCIM, the 3GPP technical report proposed three different options: a centralised SCIM, a distributed SCIM and a hybrid SCIM.

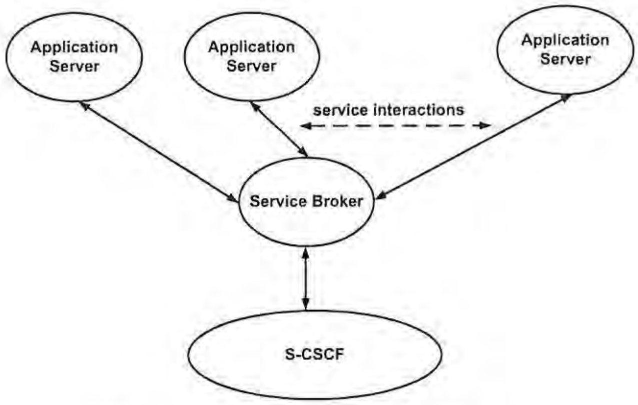


Figure 7.5: Central Service Broker.

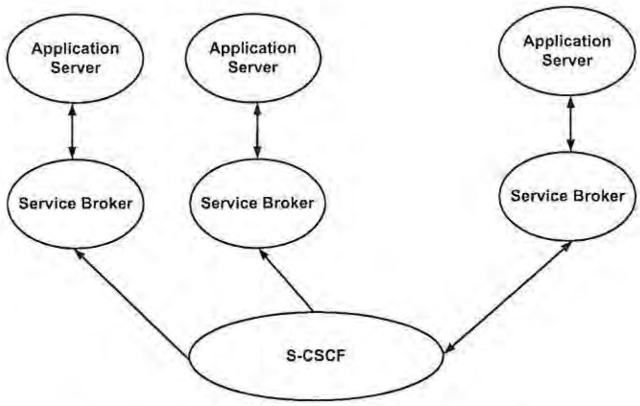
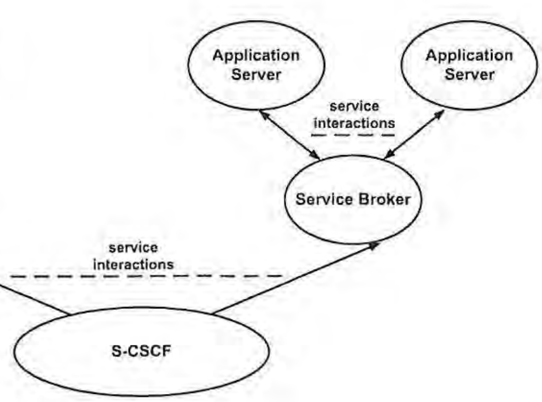
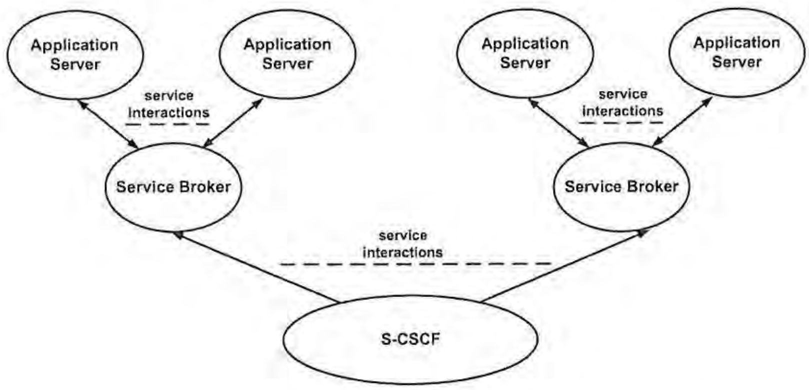


Figure 7.6: Distributed Service Broker.



(a) Version 1



(b) Version 2

Figure 7.7: Hybrid Service Broker.

Appendix B

Screenshots of the Mobicents Converged Demo

This appendix provides screenshots of the Mobicents Converged Demo system with video trailer streaming that was described in Chapter 5. It shows a more recent version of the system that was developed without IMS and allows a normal SIP client to connect to Mobicents directly.

Figure 7.8: Mobicents Converged Demo home page.

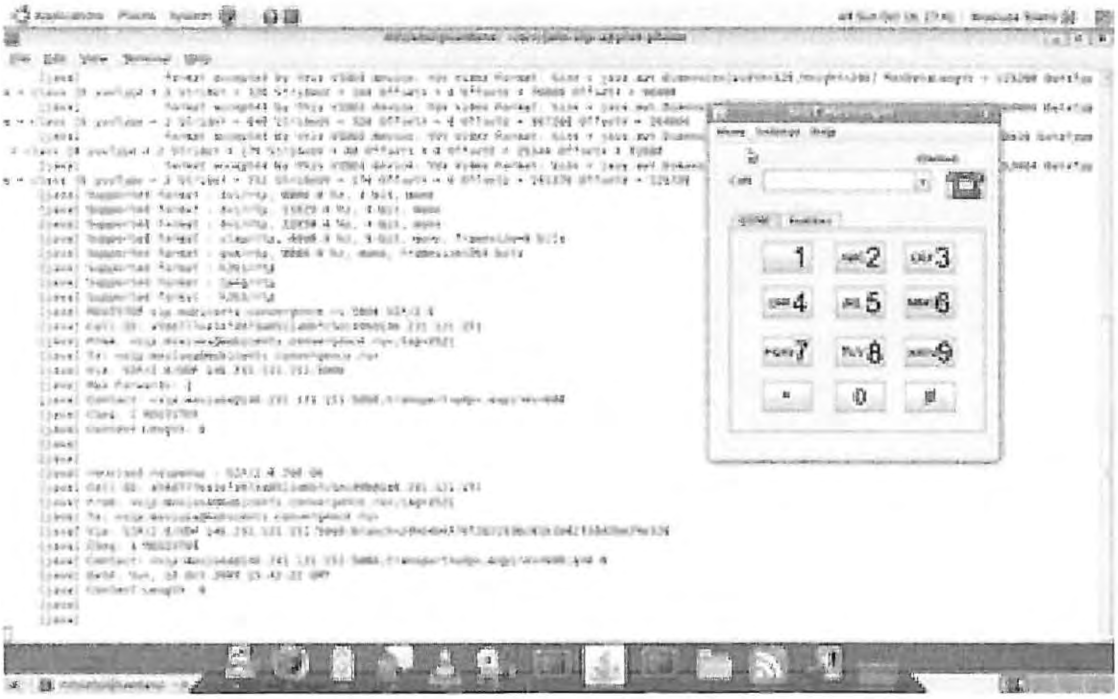


Figure 7.9: JAIN SIP Applet phone registering with Mobicents SIP application server.

Figure 7.10: Video playback troubleshooting panel on the JAIN SIP Applet Phone.

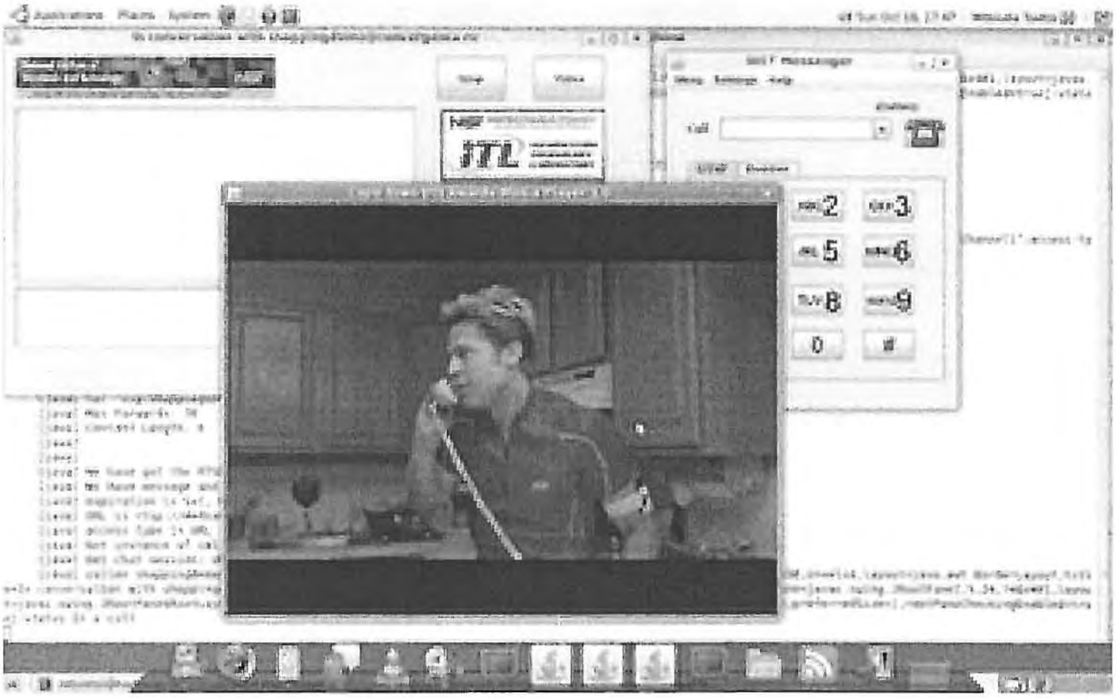


Figure 7.11: Video playback of trailer during a live session.

Appendix C

Configuration files

This appendix provides the configuration files that were used in the experiments discussed in Chapter 6. These are the DNS zone file for the open-ims.test domain, the simservs XML schema and the sip.xml file for the SCIM application in Mobicents SIP Servlets.

C.1 DNS Zone file

```
$ORIGIN open-ims.test.
$TTL 1W @
                        1D IN SOA        localhost. root.localhost. (
                        2006101001     ; serial
                        3H             ; refresh
                        15M            ; retry
                        1W             ; expiry
                        1D )           ; minimum

                        1D IN NS               ns ns
                        1D IN A          146.231.122.16

pcscf           1D IN A         146.231.122.16
_sip.pcscf      1D SRV 0 0 4060 pcscf
_sip._udp.pcscf 1D SRV 0 0 4060 pcscf
_sip._tcp.pcscf 1D SRV 0 0 4060 pcscf

icscf       1D IN A         146.231.122.16
_sip        1D SRV 0 0 5060 icscf
_sip._udp   1D SRV 0 0 5060 icscf
_sip._tcp   1D SRV 0 0 5060 icscf
```

```
open-ims.test. 1D IN A        146.231.122.16
open-ims.test. 1D IN NAPTR 10 50 "s" "SIP+D2U"  "" _sip._udp
open-ims.test. 1D IN NAPTR 20 50 "s" "SIP+D2T"  "" _sip._tcp

scscf           1D IN A        146.231.122.16
_sip.scscf      1D SRV 0 0 6060 scscf
_sip._udp.scscf 1D SRV 0 0 6060 scscf
_sip._tcp.scscf 1D SRV 0 0 6060 scscf

trcf            1D IN A        146.231.122.16
_sip.trcf       1D SRV 0 0 3060 trcf
_sip._udp.trcf  1D SRV 0 0 3060 trcf
_sip._tcp.trcf  1D SRV 0 0 3060 trcf

bgcf            1D IN A        146.231.122.16
_sip.bgcf       1D SRV 0 0 7060 bgcf
_sip._udp.bgcf  1D SRV 0 0 7060 bgcf
_sip._tcp.bgcf  1D SRV 0 0 7060 bgcf

mgcf            1D IN A        146.231.122.16
_sip.mgcf       1D SRV 0 0 8060 mgcf
_sip._udp.mgcf  1D SRV 0 0 8060 mgcf
_sip._tcp.mgcf  1D SRV 0 0 8060 mgcf

hss             1D IN A        146.231.122.16

ue              1D IN A        146.231.122.16

presence        1D IN A        146.231.122.16

mobicents               1D IN A        146.231.122.16
_sip.mobicents  1D SRV 0 0 5080 mobicents
_sip._udp.mobicents    1D SRV 0 0 5080 mobicents
_sip._tcp.mobicents    1D SRV 0 0 5080 mobicents

pcrf            1D IN A                146.231.122.16
clf             1D IN A                146.231.122.16
xcap            1D IN A                146.231.121.151
```

C.2 Simservs XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:ocp="urn:oma:xml:xdm:common-policy"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

    <!-- import common policy definitions -->
    <xs:import namespace="urn:ietf:params:xml:ns:common-policy" schemaLocation="common-policy.
        xsd"/>

    <!-- import OMA common policy extensions -->
    <xs:import namespace="urn:oma:xml:xdm:common-policy" schemaLocation="oma-common-policy.xsd
        "/>

    <!-- The element "simservs" maps to the Common Parts of a supplementary services document
        -->
    <xs:element name="simservs">
        <xs:annotation>
            <xs:documentation>XML Schema for data manipulation of Supplementary Services</
                xs:documentation>
        </xs:annotation>

        <xs:complexType>
            <xs:sequence>
                <xs:element ref="ss:absService" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="extensions" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:any namespace="##other" processContents="lax" minOccurs=
                                "0" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="absService" abstract="true" type="ss:simservType"/>
    <xs:complexType name="simservType">
        <xs:attribute name="active" type="xs:boolean" use="optional" default="true" />
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="provisioned-type">
        <xs:attribute name="provisioned" type="xs:boolean" use="optional" default="true" />
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
```

```
<xs:complexType name="supported-media-type">
     <xs:choice>
          <xs:element name="all-media" type="ss:empty-element-type"/>
          <xs:element name="no-media" type="ss:empty-element-type"/>
          <xs:sequence maxOccurs="unbounded">
               <xs:element name="media" type="ss:media-type"/>
          </xs:sequence> <xs:any namespace="##other" processContents="lax"/>
     </xs:choice>
</xs:complexType>
<xs:complexType name="provisioned-target-type">
     <xs:choice>
          <xs:element name="any-target-type" type="ss:empty-element-type"/>
          <xs:element name="telephony-type" type="ss:empty-element-type"/>
          <xs:any namespace="##other" processContents="lax"/>
     </xs:choice>
</xs:complexType>

<xs:element name="terminating-identity-presentation-restriction" substitutionGroup="
    ss:absService">
     <xs:annotation>
          <xs:documentation>Terminating Identity presentation Restriction</
              xs:documentation></xs:annotation>
     <xs:complexType>
          <xs:complexContent>
               <xs:extension base="ss:simservType">
                    <xs:sequence>
                         <xs:element name="default-behaviour" default="presentation-
                             restricted" minOccurs="0">
                              <xs:simpleType>
                                   <xs:restriction base="xs:string">
                                        <xs:enumeration value="presentation-restricted"/>
                                        <xs:enumeration value="presentation-not-restricted
                                            "/>
                                   </xs:restriction>
                              </xs:simpleType>
                         </xs:element>
                    </xs:sequence>
               </xs:extension>
          </xs:complexContent>
     </xs:complexType>
</xs:element>

<xs:element name="terminating-identity-presentation" type="ss:simservType"
    substitutionGroup="ss:absService">
     <xs:annotation>
          <xs:documentation>Terminating Identity Presentation</xs:documentation>
     </xs:annotation>
</xs:element>

<xs:element name="originating-identity-presentation-restriction" substitutionGroup="
    ss:absService">
     <xs:annotation>
```

```
        <xs:documentation>Originating Identity presentation Restriction</
            xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ss:simservType">
                <xs:sequence>
                    <xs:element name="default-behaviour" default="presentation-
                        restricted" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="presentation-restricted"/>
                                <xs:enumeration value="presentation-not-restricted
                                    "/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>

<xs:element name="originating-identity-presentation" type="ss:simservType"
    substitutionGroup="ss:absService">
    <xs:annotation>
        <xs:documentation>Originating Identity Presentation</xs:documentation>
    </xs:annotation>
</xs:element>

<!-- communication diversion rule set based on the common policy rule set.-->
<xs:element name="communication-diversion" substitutionGroup="ss:absService">
    <xs:annotation>
        <xs:documentation>This is the communication diversion configuration document.</
            xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ss:simservType">
                <xs:sequence>
                    <!-- add service specific elements here-->
                    <xs:element ref="cp:ruleset" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        <!-- service specific attributes can be defined here -->
        </xs:complexContent>
    </xs:complexType>
</xs:element>

<!-- communication diversion specific extensions to IETF common policy actions-->
<xs:element name="forward-to" type="ss:forward-to-type"/>
```

```
<!-- communication diversion specific type declarations -->
<xs:complexType name="forward-to-type">
        <xs:sequence>
        <xs:element name="target" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
        <xs:element name="notify-caller" type="xs:boolean" default="true" minOccurs="0"/
            >
        <xs:element name="reveal-identity-to-caller" type="xs:boolean" default="true"
            minOccurs="0"/>
        <xs:element name="notify-served-user" type="xs:boolean" default="false"
            minOccurs="0"/>
        <xs:element name="notify-served-user-on-outbound-call" type="xs:boolean" default
            ="false" minOccurs="0"/>
        <xs:element name="reveal-identity-to-target" type="xs:boolean" default="true"
            minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<!-- incoming communication barring rule set based on the common policy rule set.-->
<xs:element name="incoming-communication-barring" substitutionGroup="ss:absService">
    <xs:annotation>
        <xs:documentation>This is the incoming communication barring configuration
            document.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ss:simservType">
                <xs:sequence>
                    <!-- add service specific elements here-->
                    <xs:element ref="cp:ruleset" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        <!-- service specific attributes can be defined here -->
        </xs:complexContent>
    </xs:complexType>
</xs:element>

<!-- outgoing communication barring rule set based on the common policy rule set.-->
<xs:element name="outgoing-communication-barring" substitutionGroup="ss:absService">
    <xs:annotation>
        <xs:documentation>This is the outgoing communication barring configuration
            document.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ss:simservType">
                <xs:sequence>
                    <!-- add service specific elements here-->
                    <xs:element ref="cp:ruleset" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        <!-- service specific attributes can be defined here -->
        </xs:complexContent>
```

```
        </xs:complexType>
    </xs:element>

    <!-- communication barring specific extensions to IETF common policy actions-->
    <xs:element name="allow" type="ss:allow-action-type"/>

    <!-- communication barring specific type declarations -->
    <xs:simpleType name="allow-action-type" final="list restriction">
        <xs:restriction base="xs:boolean"/>
    </xs:simpleType>

    <!-- service specific IETF common policy condition elements-->
    <xs:element name="anonymous" type="ss:empty-element-type"/>
    <xs:element name="presence-status" type="ss:presence-status-activity-type"/>
    <xs:element name="media" type="ss:media-type"/>
    <xs:element name="communication-diverted" type="ss:empty-element-type"/>
    <xs:element name="rule-deactivated" type="ss:empty-element-type"/>
    <xs:element name="not-registered" type="ss:empty-element-type"/>
    <xs:element name="busy" type="ss:empty-element-type"/>
    <xs:element name="no-answer" type="ss:empty-element-type"/>
    <xs:element name="not-reachable" type="ss:empty-element-type"/>
    <xs:element name="roaming" type="ss:empty-element-type"/>
    <xs:element name="international" type="ss:empty-element-type"/>
    <xs:element name="international-exHC" type="ss:empty-element-type"/>

    <!-- service specific type declarations -->
    <xs:simpleType name="media-type" final="list restriction">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="presence-status-activity-type" final="list restriction">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:complexType name="empty-element-type"/>
</xs:schema>
```

C.3 InteractionManagerApp sip.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<sip-app>
        <app-name>InteractionManagerApp</app-name>
        <display-name>InteractionManagerServlet</display-name>
        <servlet-selection>
                <servlet-mapping>
                        <servlet-name>OfflineServlet</servlet-name>
                        <pattern>
                                <equal ignore-case="false">
                                        <var>request.method</var>
                                        <value>REGISTER</value>
```

```
                                </equal>
                        </pattern>
                </servlet-mapping>
                        <servlet-mapping>
                                <servlet-name>OnlineServlet</servlet-name>
                                <pattern>
                                        <equal ignore-case="false">
                                                <var>request.method</var>
                                                <value>INVITE</value>
                                        </equal>
                                </pattern>
                        </servlet-mapping>
                </servlet-selection>

        <context-param>
                <param-name>xdms.provider.url</param-name>
                <param-value>jnp://146.231.121.151:1099</param-value>
        </context-param>

        <context-param>
                <param-name>xdms.ip.address</param-name>
                <param-value>146.231.121.151</param-value>
        </context-param>

        <context-param>
                <param-name>xdms.port</param-name>
                <param-value>8080</param-value>
        </context-param>

        <context-param>
                <param-name>xdms.xcap.root</param-name>
                <param-value>/mobicents</param-value>
        </context-param>

        <context-param>
                <param-name>username</param-name>
                <param-value>sip:mobicents@open-ims.test</param-value>
        </context-param>

        <context-param>
                <param-name>password</param-name>
```

```
        <param-value>mobicents</param-value>
    </context-param>

    <context-param>
        <param-name>originating-identity-presentation</param-name>
        <param-value>sip:146.231.122.16:5085</param-value>
    </context-param>

    <context-param>
        <param-name>originating-identity-presentation-restriction</
            param-name>
        <param-value>sip:146.231.122.16:5090</param-value>
    </context-param>

    <context-param>
        <param-name>incoming-communication-barring</param-name>
        <param-value>sip:146.231.122.16:5095</param-value>
    </context-param>

    <context-param>
        <param-name>outgoing-communication-barring</param-name>
        <param-value>sip:146.231.122.16:5100</param-value>
    </context-param>
</sip-app>
```

Appendix D

Accompanying CD-ROM

This submission includes a CD-ROM that contains:

- Software
  - Mobicents Converged Demo
    - Source code for the Converged Demo
    - Source code for the JAIN SIP Applet Phone
    - JBoss application server with preloaded binaries of the Converged Demo
    - Sample video trailers
    - A video tutorial of the system
  - Service Repositories
    - Source code for the `SimServsAppusage` class
    - Source code for the XDMS servlets: `SimServersGlobalDocumentServlet` and `SimServsUserDocumentServlet`
    - JBoss application server with preloaded binaries for the SIP Presence Service and war files of the XDMS servlets
  - SCIM Application Server
    - Source code for the servlets that provide the SCIM: `OfflineManagerServlet` and `OnlineManagerServlet`
    - JBoss application server with a preloaded binary of the SCIM application
- Published articles that resulted from this and other related work
- Thesis document in PDF format

Appendix E

List of Publications

This appendix lists the publications that resulted from this thesis and related work:

- M Tsietsi, A Terzoli, G Wells. Using JAIN SLEE as an Interaction and Policy Manager for Enabler-Based Services in Next Generation Networks. International Journal of Electronics and Telecommunications, Vol. 56, No. 2, pp 117–124. June, 2010.
- M Tsietsi, A Terzoli, G Wells. Towards Service Capability Interaction Management in IMS Networks. In SATNAC '10: Proceedings of the 13th Southern African Telecommunications Networks and Applications Conference. Stellenbosh, South Africa. 5 to 8 September, 2010.
- M Tsietsi, R Musvibe, A Terzoli, G Wells. Towards IPTV Service Discovery and Selection in an IMS environment. In ICUMT '10: 2nd International Congress on Ultra Modern Telecommunication and Control Systems, pp 195–201. Moscow, Russia. 18 to 20 Oct, 2010.
- M Tsietsi, A Terzoli, G Wells. JAIN SLEE as a Service Capability Interaction Manager for the IP Multimedia Subsystem. In BROADBANDCOM '09: 4th International Conference on Broadband Communication, Information Technology and Biomedical Applications, Wroclaw, Poland. 15 to 18 July, 2009. ISBN: 2009, 978-83-7493-470-1.
- M Tsietsi, Z Shibeshi, A Terzoli, G Wells. A Telephony Driven Framework for e-Learning Using Open Protocols and Open Source Software. In ITU-T Kaleidoscope '09: Academic Conference on Innovations for Digital Inclusion, pp 149 –153. Mar Del Plata, Argentina. 31 August, 2009 to 1 September, 2009.

- M Tsietsi, A Terzoli, G Wells. Mobicents as a Service Deployment Environment for Open IMS Core. In SATNAC'09: Proceedings of the 12th Southern African Telecommunications Networks and Applications Conference. Ezulwini, Swaziland. 30 August to 2 September, 2009.
- M Tsietsi, A Terzoli, G Wells. An Open Service Delivery Platform for Adding Value to Softswitch-based Telephony Environments. Work in progress. In SATNAC'08: Proceedings of the 11th Southern African Telecommunications Networks and Applications Conference. Stellenbosch, South Africa. 7 to 10 September, 2008.