Abstract—The composition of large systems from smaller building blocks is a common system design strategy. It makes the process of introducing changes to such systems more economical and also supports the rapid development of entirely new systems. This technique is now being applied in telecommunication systems such as the IP Multimedia Subsystem (IMS) whereby standards development organisations have been actively encouraging network operators to build telecommunication services from basic components. In order to coordinate the interactions that must occur in a multi-component service, an IMS node called the Service Capability Interaction Manager (SCIM) has been proposed. Though the SCIM is central to realising the benefits of service decomposition, it has not yet been formally incorporated into the set of standards that constitute the IMS. This has left issues regarding the functional and structural architecture of the SCIM open to interpretation. This paper outlines the authors’ own interpretation of the architecture of the SCIM that stems from preliminary work initiated by the Third Generation Partnership Project (3GPP) as well as contemporary proposals by researchers in the field. The development of a prototype is described that has been built using a servlet container that supports the creation of converged SIP/HTTP applications. This investigation is part of a larger project aimed at providing an Extended IMS Service Layer (EISL) that is more conducive to the development and deployment of complex multimedia services in an IMS environment.

Index Terms—IMS, SCIM, Service Broker, Servlets, Mobicents

I. INTRODUCTION

The IP Multimedia Subsystem (IMS) is middleware in a 3G mobile system that acts as a routing substrate that conveys application requests to a service layer that is populated with application servers. As such, the IMS does not directly host the services but it provides inbound and outbound communications with security, quality of service (QoS) and policy guarantees, and supplies developers with standard interfaces for docking their applications.

Developers are usually afforded the freedom to use whatever techniques they wish to develop services for the network as long as they use standard interfaces to connect to the IMS. More recently, the standards development organisations behind the specification of the IMS have attempted to provide developers with a framework which they believe will help ensure network operators are able to provide a sustainable and thriving multimedia services platform. One of these bodies, the

Third Generation Partnership Project (3GPP), states in [1] that their objective is not to standardise the full set of services that operators can provide, but rather their aim is to standardise service capabilities which are the basic mechanisms needed to realise services. The 3GPP anticipated that this move would aid the rapid development of telecommunication services and would ensure interoperability and interworking among networks.

While the benefits are clear, the use of service capabilities for constructing services challenges the way in which the IMS layer currently operates. Typically the IMS routers (call session control functions) are able to use network-hosted information to map a request to a designated application server that will handle that request. This is fairly easy to do since requests are carried using SIP (Session Initiation Protocol) which standardises the structure of requests and makes it possible to cater in advance for certain request types. The distribution of service intelligence across several nodes means that these existing methods alone are no longer sufficient for this purpose.

This has led to the definition of a node called the Service Capability Interaction Manager (SCIM) which has been identified by the 3GPP as the node responsible for managing the interactions between services and hiding this complexity from the core IMS [2]. The 3GPP has not yet fully standardised the SCIM and much of what is understood about it has been contributed by the research community.

This paper describes a proposal for the structural and functional architecture of a SCIM that is based on an analysis of contemporary literature. This specification is underpinned by two important design goals which are to ensure that the impact of the introduction of the SCIM on standard IMS entities is minimised and that existing IMS protocols and mechanisms must be reused wherever possible. A discussion is also provided on the implementation of a prototype that was performed using a SIP Servlet container from an open source project called Mobicents.

This paper is organised as follows: Section II presents a review of pertinent literature from the 3GPP and the broader research community. Section III introduces the design of an extended IMS service layer that incorporates a SCIM and other supporting infrastructure. Section IV takes a detailed look at the proposed structure of the SCIM and shows how its modules
II. Research Context


The 3GPP conducted a preliminary investigation into the possible impacts of introducing a SCIM into the IMS network in 2008. This work resulted in the publication of a technical report that summarises some of the key challenges, open questions and possible solutions [3].

The bulk of the report examines issues concerning how the SCIM performs its role of interaction management in an environment where services are built from multiple service capabilities. The report maintains that existing methods which are based on static rules (called initial filter criteria, or iFC) are not well suited for invoking services in a distributed environment. In addition to this, the 3GPP identified other scenarios in which existing methods would not be able to suffice. For instance, these methods are not able to cater for dynamic conditions that cannot be expressed using the SIP protocol. Examples include the use of the time of day or the use of calendar events such as anniversaries to influence the delivery of a service. Secondly, providing service personalisation for both user (in the form of preferences) and the operator (in the form of application level policies) was also identified as a feature that could increase revenue, but static methods cannot easily be used to achieve this either. In addition to these, the 3GPP also recognised the potential for conflicts to arise due to undesirable interactions that could occur between service capabilities. This concern is valid because while it is easy to anticipate the results of the execution of a single entity, it is more difficult to anticipate the execution of multiple entities executing one after the other in a composite service.

To solve these problems, the 3GPP proposed the creation of offline and online interaction management techniques. Offline interaction management refers to mechanisms that are employed to provision rules into the system that will be used to influence the execution of a user request. Online interaction management depends on the offline stage and refers to the techniques executed by the SCIM during a live session. It was anticipated that the SCIM would reuse iFC provisioned in the user profile to create a chain of service capabilities to execute that could be interleaved based on dynamic conditions, user preferences and operator policies. This chain would be used in concert with an historical vector that contains the identities of the service capabilities that have already been executed in order to influence service invocation based on previously executed nodes. To ensure invalid interactions, information specifying legal interactions would be incorporated into the Home Subscriber Server (HSS) during the offline interaction management stage. To perform all this, the SCIM would need a Diameter interface in order to download such rules from the HSS and a SIP interface with the application servers as well as the serving call session control function (S-CSCF).

There has been no follow up to this technical report and as such practical ways in which some of these processes could be implemented have never been fully specified by the 3GPP.

B. Contemporary Proposals

The IMS research community identified this gap in knowledge and has contributed its own proposals that attempt to describe the SCIM in more detail. Some of these proposals have addressed basic routing mechanisms and the relationship that the SCIM has with the S-CSCF. In [4] the authors suggest that the SCIM must work closely with the S-CSCF to invoke service capabilities. Rules that indicate how services are composed from different service capabilities would be carried in a service capability profile (SCP) that would be stored in the HSS, further emphasising the need for the SCIM to have a Diameter interface with the HSS. Offline interaction management techniques would be used to make this information available to the SCIM.

Similar proposals can be found in [5] and [6]. These differ with [4] in that they decompose the SCIM into a distributed system that consists of a central node and auxiliary nodes. This change alleviates the S-CSCF from active participation allowing the SCIM to accomplish routing in collaboration with the application servers hosting the services. To further explain how offline interaction management works, in [6] the authors propose that the registration of a user should be used as a signal for the SCIM to download interaction rules into memory.

These three proposals address some portions of the requirements, but do not address methods for the avoidance of undesirable interactions or service personalisation. In [7], the authors present their model of an interaction manager that caters for these features. A feature interaction detection and resolution (FID& R) module is embedded into the SCIM that maintains a record of all invalid interactions. Another module called the Favourite AS (application server) list contains a list of user defined application servers that the user favours for use during the delivery of a service. Researchers at the FOKUS Fraunhofer institute have specified a similar architecture in [8]. In this paper, the authors describe a node called a service broker which contains a SCIM inside of it. The service broker contains other components in addition to the SCIM. A service registry contains a list of all known services offered by the network and policy evaluation, policy enforcement and policy management components deal with operator policies that assist in the avoidance of undesirable interactions.

III. The EISL System

The literature that has been reviewed provides a rough sketch of the SCIM. However, if it is to be incorporated, it will become part of a larger ecosystem and must be able to interact with existing nodes. There have been proposals that have motivated for the modification of standard IMS interfaces in order to cater for interaction management. For example, the 3GPP technical report suggests that interventions in the existing iFC could be made to help cater for dynamic or non-SIP conditions such as user agent capabilities. It further makes
the suggestion that the SCIM could possibly be integrated with the S-CSCF as an additional module. The authors in [4] also state that the modification of the existing HSS data model is necessary in order to cater for their SCP data structure. The SCP would be integrated into the existing data model, thus changing the standard design.

While these changes may assist in addressing the problems that need to be solved, interventions of this nature are not ideal as they introduce changes to standardised features of the IMS which can prove to be costly to implement in existing networks. A more suitable course of action would be to align designs in such a way that they introduce as few changes as possible to the existing system. This includes the reuse of existing IMS protocols and procedures wherever possible. What is required is the specification of an IMS service layer that fulfills these criteria.

EISL stands for the Extended IMS Service Layer, and it defines a solution architecture that uses existing IMS protocols and does not impose any changes to standard IMS nodes [9]. EISL is depicted in Figure 1. A comprehensive account of the features of the EISL system is provided in [10]. The next few sections describe the main components and users of EISL followed by a detailed account of the structural and functional architecture of the SCIM.

A. The Network Administrator

The administrator is responsible for carrying out tasks defined by the network operator. This includes those tasks that are already performed in existing networks such as the provisioning of new users and the management, deployment, configuration and monitoring of application servers. The administrator must also perform tasks that are mandated by EISL such as the provisioning of service composition information and the specification of operator policies on service interactions.

B. Service Broker / SCIM

The service broker is a complex node that encapsulates the SCIM among other modules [8]. It is anticipated that, in addition to providing the SCIM, it will also assist in servicing requests from third parties on behalf of the network. This includes servicing discovery requests that probe for underlying service capabilities and executing policies that define the appropriate use of service capabilities by such third parties.

The SCIM has similar interfaces to a standard IMS application server, which are the ISC (Internet protocol multimedia Service Control) interface with the S-CSCF which is implemented in SIP and the Sh Diameter interface with the HSS. The SCIM also has an interface with service repositories which it uses during online sessions and while in standby mode. During an online session, the SCIM obtains service compositions in order to execute a user request. In the offline stage, it facilitates service personalisation on behalf of the user.

C. Service Repositories

The service repositories store service related information, much like the service registry in [8]. EISL defines two types of repositories which are the service repository and the user service repository. The service repository stores information about how complex services are composed from service capabilities, much like the SCP described in [4]. This repository is created and managed by the network administrator. Each service must be assigned a unique identifier and multiple service capabilities can be inserted under a single service. Service capabilities belonging to a service are assigned priority numbers so that the SCIM can know the appropriate invocation precedence order.

The second repository is very similar in structure to the first one and in fact inherits content from it. Content is copied into the user space in the user service repository after which...
the user can proceed to modify and personalise the contents. Each service is expected to indicate legal composition combinations which the user is allowed to modify. In addition, personalisation allows users to specify conditions that can be used to modify the behaviour of a service during runtime by using parameters such as time of day, calendar events, presence status or even the result of a previously executed service.

As the underlying technology, the service repositories can be implemented in the HSS as some proposals provide for, but the use of an XML Document Management Server (XDMS) is recommended in EISL. 1 An XDMS can store service information in the form of XML documents which allows client devices to issue requests to it using the XML Configuration Access Protocol (XCAP). The XDMS is a standard feature of the presence architecture in IMS and the XCAP protocol is a standard feature of an IMS user agent.

D. Service Interactions

Information about the invocation order for composed services is provided in the service repositories. In a similar fashion, it is possible to provide policies in the service repositories that adjust the precedence order in order to avoid certain interactions that could result from a service chain. In addition, it is also necessary to define appropriate behaviours under situations that violate interaction rules. Since these policies apply globally, the service repository is the most appropriate place to insert them so that they can be absorbed into user-space configurations.

E. Service Personalisation

Service personalisation in EISL is catered for in two modes in EISL: via direct interaction and via the SCIM as an XCAP proxy. The direct interaction method allows the IMS user agent to issue XCAP requests directly towards the user service repository. The drawback with this method is that it requires the user agent to be able to switch between the two service repositories depending on the existence or non-existence of personalised information. When the SCIM is used as an XCAP proxy, the user agent need only be aware of the user service repository and the SCIM mediates between the service repositories and the user agent. This method also supports the reduction of bandwidth consumption on the access network since the bulk of the interactions are carried by the SCIM, which resides in the bandwidth-rich IMS core.

IV. THE SCIM IN DETAIL

The SCIM is able to operate within the EISL ecosystem through standard protocol interfaces. This section examines the internal structure of the SCIM and provides a discussion on how these internal components work together to fulfill its goals.

1 The recommendation of the use of an XDMS over the HSS is motivated in [10], which compares the two technologies for this particular use case.

A. SCIM Structure

Figure 2 shows the structural architecture of the SCIM. The SCIM possesses three protocol stacks which are SIP, Diameter and XCAP. The SIP stack interacts with the S-CSCF and service capabilities during online interaction management and the Diameter and XCAP stacks interact with the HSS and the XDMS respectively during offline interaction management.

![Figure 2. Structural architecture of the SCIM.](image)

The SCIM requires a local storage space to store information that it needs to interact with EISL entities. The Configuration Registry fulfills this requirement. It is provisioned with information by the administrator. For example, the registry can be provided with the network addresses of the HSS or the XDMS in order to allow the Diameter and XCAP stacks to properly format their requests. The Composition Engine is responsible for converting rules and composition information from either Diameter or XDMS sources into a homogeneous format that can be inserted into the Composition Rules Cache. The latter stores this information in a way that is easy for the SIP portion of the SCIM to use for subsequent invocations.

B. SCIM Functions

The main functions of the SCIM are to manage service interactions and to perform both types of interaction management. These processes are described next.

1) Management of Service Interactions: Information about the legal compositions and interactions between services is contained in the service repositories. Thus the SCIM need only to download this information into its cache and interrogate them during online interaction management. In addition, the registry can be used to store a historical vector of invoked services in order to decide whether or not to invoke a specific service by interrogating operator policies in the service repository.

2) Offline Interaction Management: Offline interaction management is catered for using either Diameter or XCAP methods. When Diameter is used, the interaction rules must be uploaded from the HSS. This can occur when a user registers or if the service profile of a registered user is changed. Since this must be accomplished using the Sh Diameter interface, the
most appropriate commands to be used for this are the User Data Request (UDR) and Subscription Notification Request (SNR) commands respectively, since the UDR command is used to request user data and the SNR command is used to subscribe to changes in user data.

The second and recommended method uses XCAP. Interaction rules are still downloaded when a user registers if the SCIM has a third party registration contract with the IMS. This can be achieved using existing iFC techniques. When the registration is received, the SCIM examines the cache for interaction rules for this user. If they do not exist, it issues an HTTP GET and downloads them from the XDMS. It passes them to the Composition Engine which converts the rules from an XML format into one that can be appropriately stored in the cache.

3) Online Interaction Management: A SIP request reaches the SCIM through iFC configurations. The SCIM examines the Composition Rules Cache to extract the rules and compositions that apply to a request. As long as the third party registration is in effect, the SCIM should have this information on hand. If not, it can execute a download during the live session using the techniques described in the previous section.

The SCIM obtains the identity of the service capability with the highest priority and examines the policies that relate to it. Service interaction rules take precedence and the SCIM first tries to determine if the invocation of that service capability violates any policies. If not, the SCIM next examines user preferences to determine if any violations will occur. If none of these conditions are true, the SCIM invokes the service and once successfully executed, stores its identity in a vector. It repeats this process for all service capabilities in the service.

V. PROTOTYPING A SOLUTION

EISL defines an infrastructure that is based on standard IMS protocols and procedures. This property made the development of a prototypical testbed much easier to perform. Notably, the FOKUS Open IMS Core project [11] could be used in its unmodified state to provide the call session control functions and the HSS without having to make interventions in the source code. For the SCIM, it was evident that a platform was needed that could provide SIP, Diameter and XCAP interfaces. Of the currently available open source platforms, the Mobicents project was found to be the most suitable for modeling the SCIM.

Mobicents is the umbrella name for a set of application servers that provide multimedia services using standard Internet and IMS protocols. One of these is Mobicents SIP Servlets which provides a servlet container that is compliant with the SIP servlets version 1.1 specification as defined in [12]. It provides the necessary tools for developing the SCIM since it supports the creation of converged SIP/HTTP applications which caters for the SIP and XCAP related functions. In addition, Mobicents Diameter, a sister project to Mobicents SIP Servlets, provides the Diameter libraries that can be used for the Sh interface. However, since the HSS is not the recommended location for the storage of interaction details, this support was not explored in the experiment.

A. Structure of the Prototype

Figure 3 shows the structure of the experimental testbed.

The servlet container is deployed onto a JBoss application server and hosts a servlet application called InteractionManagerApp. The servlet application consists mainly of two servlets: OfflineManagerServlet which handles the offline interaction management functions and OnlineManagerServlet which handles the online functions. The ETSI simservs application usage was used which defines service compositions involving simulation services [13]. Simulation services are basic services that include call barring, call hold and call diversion. Therefore, an additional servlet, ClientSimulatorServlet was created in order to simulate interactions with the XDMS since the authors were not able to find an IMS client that was compatible with the XCAP application usage that was used in the XDMS. The InteractionManagerApp has a servlet context which allows the servlets to store persistent data.

B. Implementing Offline Interaction Management

The Open IMS Core project (i.e. FHoSS + xCSCF) provides developers with the ability to create third party registrations through iFC configurations that modify a user’s service profile. One such registration was created for a user Bob. When Bob’s SIP REGISTER request is received by the SCIM, it examines the Configuration Rules Cache for information about this user’s service compositions and rules. This cache is implemented in a servlet application context which is a block of memory provided by the servlet container. If the cache is empty, it issues an XCAP GET request towards the XDMS to download the rules for this user. Once obtained, it stores this information for future use. The actual data is stored in a HashMap data structure which stores a list of services in the form <Priority,ServiceName>. The same IMS configuration also supported the removal of such records when the user deregisters.

C. Implementing the Service Capabilities

Since the XCAP application usage for the XDMS defines simulation services, it was decided that the service capabil-
ities would be modeled as simulation services as well. In an experiment, three simulation services were used which implemented call barring (incoming and outgoing) and an originating identity service. The simulation services were implemented as instances of a Kamailio server, which is an open source SIP server [14]. For simplicity, the services were statically configured to return certain values depending on the use case being tested. This simplified the implementation of the services and was meant to simulate the condition of a service capability evaluating a policy and deciding whether or not to deliver a service to the user or not.

D. Implementing Online Interaction Management

When such a message is received by the container, the OnlineManagerServlet examines the request and looks up the next service to invoke. It obtains this information from the Configuration Registry which is fed service name to IP address mappings from the sip.xml deployment descriptor file in  

<context-param> tags. This information is loaded into the application context when the servlet application is deployed. Each time that the servlet application successfully dispatches a SIP request to a service capability, it puts the service name into a data structure and inserts it at runtime into the servlet application context. As such, the servlet application context is useful for loading and unloading static information and dynamic information to be shared between the servlets during its execution. The ability to store and retrieve state is important since SIP servlets are stateless SIP entities and would not be able to perform SCIM duties without some form of support in this regard.

VI. CONCLUSION AND FUTURE WORK

This paper describes a proposal for the structural and functional architecture of a SCIM for IMS that exists within the context of an extended IMS service layer. EISL caters for the requirements of interaction management in an environment where services are implemented from service capabilities. It does so without requiring changes in existing IMS nodes and reuses standard IMS protocols and mechanisms. These properties make it appealing to network operators since the cost of embedding the SCIM would be minimal as EISL provides a SCIM that can easily be plugged into the network. The prototype that is presented was an initial implementation that was developed for proof of concept and worked for a small number of service building blocks.

In future, work will be undertaken to provide a custom application router that is able to interact appropriately with the service repositories in order to parse operator policies and user preferences in order to make appropriate decisions on how best to route requests. As the SCIM is only a component of the service broker, more work will be done to provide other functions that are generally attributed to the service broker. The SCIM that is presented here is a centralised SCIM which is easy to conceptualise. In future, distributed architectures will be investigated that avoid bottlenecks by providing a system with redundancies.

ACKNOWLEDGEMENTS

The work reported in this paper was undertaken in the Telekom Centre of Excellence in Distributed Multimedia at Rhodes University with financial support from Telkom SA, Comverse, Tellabs, Stortech, Easttel, Bright Ideas 39 and THRIP.

REFERENCES


Mosiuoa Tsietsi is currently working as a postdoctoral researcher in the department of Computer Science at Rhodes University. His research interests include open source software and service delivery platforms for IMS.

Alfredo Terzoli is Professor of Computer Science at Rhodes University, where he heads the Telkom Centre of Excellence in Distributed Multimedia. He is also Research Director of the Telkom Centre of Excellence in ICT for Development at the University of Fort Hare. His main areas of academic interest are converged telecommunication networks and ICT for development.

George Wells is Professor and Head of the Department of Computer Science at Rhodes University. He completed his PhD degree in coordination languages for parallel and distributed programming at the University of Bristol, and continues to do research in this area, with a particular interest in methods for the simple, efficient use of multicore processors.