TAMPEREEN TEKNILLINEN YLIOPISTO

MARKKU KORKEALA

System for Cross-domain Identity Management for Access Control of SOA Services

Master of Science Thesis

Examiners: Professor Kari Systä and University teacher Marko Helenius
Examiners and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 8th of June 2016

ABSTRACT

KORKEALA, MARKKU : System for Cross-domain Identity Management for Access Control of SOA Services
Tampere University of Technology
Master of Science thesis, 62 pages, 17 Appendix pages
August 2016
Master's Degree Programme in Information Technology
Major: Software systems
Examiners: Professor Kari Systä, University teacher Marko Helenius
Keywords: System for Cross-domain Identity Management, SCIM, REST, IAM, Identity and Access Management, IdM, Identity Management

Identity and Access Management systems are usually fundamental services in organizations. In Service-Oriented Architecture (SOA) they can be used to provide three different services: authentication, authorization and information about users and their access rights. For the latter, there has not been a widely used standard in SOA to provide user information to other services. System for Cross-domain Identity Management (SCIM) is a new emerging Representational state transfer (REST) based standard to help provision user information to cloud services.

This Master Thesis discusses how SCIM can be used to provide user information to consuming services in a SOA based solution. The first part of the thesis studies what are the advantages and disadvantages using REST based solutions compared to SOAP based solutions. Based on a literary review, REST has better performance, measured by throughout put, and it is independent of data format. SOAP has the advantage of being very standardized and has mature tools and frameworks compared to REST. REST is more based on conventions than standards, so tools and frameworks behave differently which might lead to interoperability problems.

The second part of the thesis focuses on whether SCIM can be used to provide user information service to consuming services. Three scenarios were designed and implemented in SCIM to find out whether the access right model of the SCIM is expressive enough and whether the resources defined by SCIM provide a required set of attributes. The presented scenarios have different requirements: the first one models internal access rights of an organization, the second scenario a use case in which an organization offers services to its customers and the third one a use case in which role based access rights are restricted to certain objects. The last two scenarios required extending the SCIM core resource schema.

The models were tested in a proof-of-concept implementation and they were able to fulfill all the requirements. This indicates that SCIM can be used to implement user and user's access right information service. To conclude, a five step process is presented that an organization can use to assess if SCIM is suitable for its use.

TIIVISTELMÄ

KORKEALA, MARKKU: System for Cross-domain Identity Management palveluiden pääsynhallintaan palvelupohjaisessa arkkitehtuurissa
Tampereen teknillinen yliopisto
Diplomityö, 62 sivua, 17 liitesivua
Elokuu 2016
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastajat: Professori Kari Systä, Yliopisto-opettaja Marko Helenius
Avainsanat: System for Cross-domain Identity Management, SCIM, REST, Identiteetin- ja pääsynhallinta, IAM, IdM

Identiteetin- ja pääsynhallintajärjestelmät ovat yleensä yksi organisaation perusjärjestelmistä. Palvelupohjaisessa arkkitehtuurissa (Service-Oriented Architecture, SOA) ne voivat tarjota palveluna autentikointia, auktorisointia ja käyttäjän identiteetin tietoja. Jälkimmäisestä ei ole laajalle levinnyttä SOA-tyyppistä avointa standardia. System for Cross-domain Identity Management (SCIM) on uusi, Representational state transfer (REST)-pohjainen standardi käyttäjätiedon provisioimiseen pilvipalveluihin.

Tämä diplomityö käsittelee SCIM:n käyttämistä käyttäjien identiteettitiedon tarjoamiseen palveluna muille järjestelmille. Ensimmäinen osa työtä perustuu kirjallisuuskatsaukseen ja tarkastelee mitä hyviä puolia ja haittoja REST-pohjaisen teknologian käyttämisessä on verrattuna SOAP-pohjaiseen teknologiaan. Tutkimusten suorituskykymittauksien perusteella REST-pohjaisten ratkaisujen läpivienti on SOAP-pohjaisia parempi. REST on myös riippumaton sisällön esitysformaatista. REST-teknologian huonona puolena on standardien puute: työkalut ja kehykset tuottavat erilaisia ratkaisuja, jotka voivat lisätä yhteensopivuusongelmia.

Tutkimuksen toinen osa käsittelee SCIM-standardin käyttämistä käyttäjätiedon ja käyttäjän käyttöoikeuksien tarjoamiseen palveluna muille sovelluksille. Tämän tutkimiseksi tässä diplomityössä kehitettiin kolme skenaariota, jotka mallinnettiin SCIM:llä. Ensimmäinen skenaario mallintaa organisaation sisäisiä käyttöoikeuksia, toinen tilannetta, jossa organisaatio tarjoaa palveluita asiakkailleen, ja kolmas esittää roolipohjaisen käyttöoikeuksien rajoittamisen käyttöoikeuteen yhdistettyihin objekteihin. Kaksi viimeisintä mallia vaativat SCIM-standardin resurssiskeemaan laajentamisen vaatimuksien täyttämiseksi.

Mallien toimivuus verifioitiin konseptitoteutuksella. Kaikki skenaarioiden vaatimukset pystyttiin toteuttamaan, minkä perusteella voidaan sanoa SCIM:n sopivan käyttäjätiedon ja käyttäjän käyttöoikeuksien tarjoamiseen palveluna. Lopuksi esitellään viisiosainen prosessi, jota organisaatiot voivat käyttää kartoittamaan SCIM:n sopivuutta niiden vaatimuksiin.

PREFACE

Ideas for the topic of this Masters Thesis started surfacing in 2013. I drafted some outlines then, but really started working on it in the autumn 2015. Since then the topic and the angle have changed quite a bit, though it has always been about System for Cross-domain Identity Management. Most of the writing and programming was completed on a study leave in spring 2016.

I would like to thank my examiners, Kari Systä and Marko Helenius. They provided valuable feedback and guided me through the writing process. They also showed patience in times when I was too busy at work to concentrate on my thesis. I am grateful for their guidance and support.

Secondly, I would like thank my employer, Nixu Oyj, for giving me the possibility to take a study leave and to work full time on my thesis. They always encouraged me to conclude my studies and also gave me ideas for the thesis.

I would also like to thank my family. They have always been supportive of me no matter what I do. A special thank you to my Kaisa. She has always supported and encouraged me on this path.

Helsinki, 2.8.2016

Markku Korkeala
markku.korkeala@iki.fi

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABAC Attribute-based access control is an authorization model, where access to resources is determined by attributes of the user.

API Application Programming Interface is a defined interface with which computer programs can use other components or services.

ESB Enterprise Service Bus is a messaging bus used in enterprises to integrate applications and services.

IAM Identity and Access Management is the term used for managing user's digital identities and access rights.

IdM Identity Management is a process to maintain information about digital identities throught their lifecycle.

IdP Identity Provider is a service provider, which provides identification of users for other service providers.

JSON JavaSript Object Notation is a text-based and language-independent data interchange format.

LDAP Lightweight Directory Access Protocol is an open protocol to exchange distributed directory information.

OASIS Advancing Open Standards for the Information Security is a consortium that develops and promotes usage of open standards.

RBAC Role-based access control is an authorization model where access to resources is determined by user's roles.

REST Representational state transfer is an architectural style of the World Wide Web.

SaaS Software as a service is a cloud computing model where customers rent rights to use software via Internet.

SAML Security Assertion Markup Language is an open XML-standard to exchange authentication and authorization information.

SCIM System for Cross-domain Identity Management is a standard for federating and connecting identities for different domains.

SOA Service-Oriented Architecture is a software architecture style where software systems provide each other services through language-agnostic API's.

SOAP Simple Object Access Protocol is a protocol to exchange XML based information.

SP Service Provider provides its users services over a network.

SPML Service Provisioning Markup Language is an XML-based language for transfering user and resource based information.

URI Uniform Resource Identifier is an identifier for a resource in a network, usually World Wide Web.

WSDL Web Services Description Language is an XML-based language to define and describe web service API's.

WSOA Web Service-Oriented Architecture is a SOA-based architecture model where services are provided by web services.

1. INTRODUCTION

Service-Oriented Architecture (SOA) is a widely used architecture style in the enterprise sector that supports service-orientation. Service-orientation means designing software systems in terms of services. In SOA a service has the following characteristics:

- It implements logical business activity
- It is used over communication protocol, usually over network
- It can be composed of multiple services
- Implementation is a black box for service consumers

One of the key principles of SOA is to build services that are independent of technology, product and vendor. Deployed services can be consumed by other software systems, making the former a service provider and the latter a consumer (see figure 1.1). The provider can also be a legacy system on which a separate wrapper component is built to provide the service interface. Services can register to an external registry from which consumers can find available providers. These registries can be public on the internet or private in the organizations internal network.

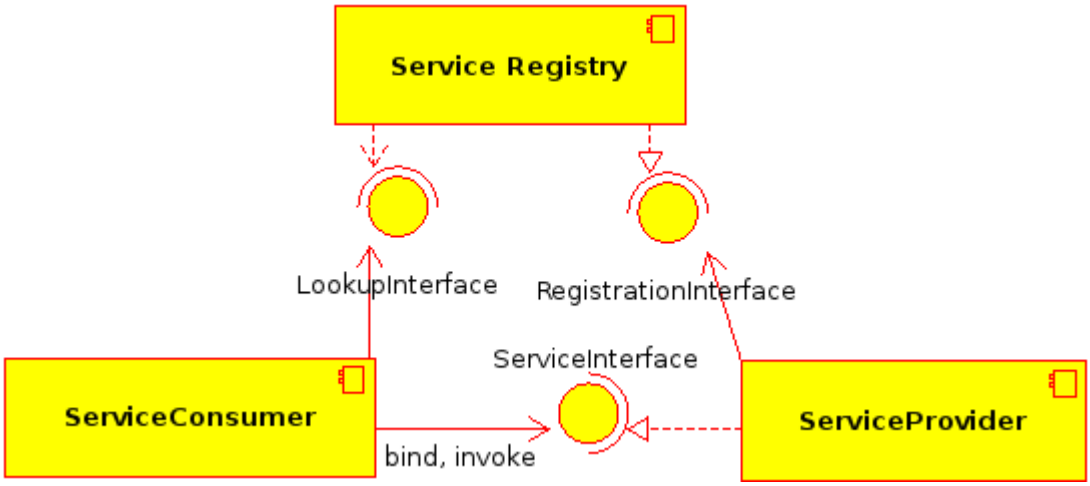

Figure 1.1: Service consumer and provider in SOA

This separation of software systems to services provides many benefits, for example it makes them autonomous and loosely coupled (Feuerlicht 2010, p. 2) and

provides business agility (Arsanjani 2004, p.4). Benefits of autonomous and loosely coupled services are that they can be developed separately from each other. Also applications can be orchestrated to use many underlying services with focus on business requirements and processes. Other added benefits of SOA include statelessness, discoverability and reusability. Usually these services are built as web services, which are accessed using Simple Object Access Protocol (SOAP) and described with Web Services Description Language (WSDL). This is referred as Web Service-Oriented Architecture (WSOA).

1.1 Service-oriented Architecture with Enterprise Service Bus

A popular component in WSOA deployments is a Enterprise Service Bus (ESB), which act as a broker; services are linked to the ESB instead of being used directly by the consumer (see figure 1.2). Consumer sends messages to ESB which then routes them to the service and sends the response from the service back to the consumer. Core tasks of the ESB are service registry and orchestration. Usually ESB has the following duties when acting as a broker: message validation, transformation, content-based routing, security, service discovery, load balancing, and logging (Groba et al. 2008, p. 7).

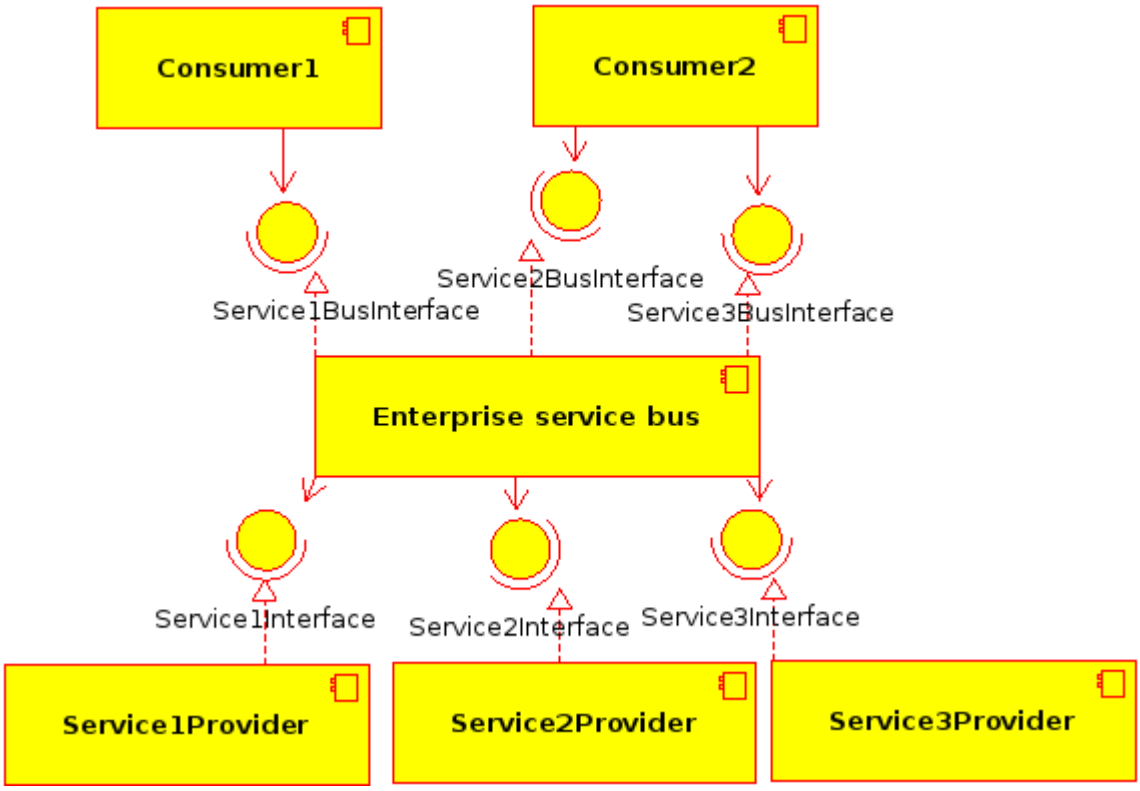

Figure 1.2: SOA architecture with ESB

1.2 Identity and Access Management

Identity and Access Management (IAM) means managing digital identities and access rights throughout their lifecycle. When organizations grow bigger and numbers of employees and computer systems grow, the management of user identities and access rights becomes a burden for the administrations. Centralized user management has been one the solutions developed for this problem. In centralized IAM other software systems use one central system to query information about users and their access rights or the user access rights information is provisioned to other software systems from the centralized IAM system. This leads to efficiency which will bring cost savings for the organization as information about users and their access rights is maintained in one place and user accounts do not need to be created separately for every system. It also helps implementing improvements on user experience, such as reducing the number of usernames for users and implementing single-sign-on.

Centralized IAM can also be seen as a security aspect for the organization: managing all the access rights for identities in a centralized system makes it easy to audit access rights, to find dangerous role combinations and to remove unnecessary access rights for users. When a user resigns from the organization, a centralized IAM solution helps revoking all the access rights of the user from different services.

The IAM system is usually seen as a mandatory service to protect necessary resources in the services of the organization. In addition to security and efficiency, a third motivation for IAM in organizations are the potential new business opportunities it provides. Investing to IAM might open new ways of operating or new revenue, for example in Finland the banks provide strong authentication as a service for third parties so that the identity of the bank customers can be verified by a third party (Linden 2009, p. 71).

1.3 Research Goals and Methods

System for Cross-domain Identity Management (SCIM)[1] is a new Representational state transfer (REST) based standard for cross-domain identity management. REST is an architecture style for the World Wide Web first introduced by Roy Fielding (Fielding 2000, 94-124). In REST, resources are identified and located by their Uniform Resource Identifier (URI). Using HTTP methods (GET, DELETE, PATCH, POST, PUT), the state of the resources can be retrieved or modified.

This thesis studies how SCIM can be used in SOA to provide identity management services for an organization in an on-premises solution. As SOAP has been in the past one of the dominant technologies, an organization might have implemented on-premises services using SOAP. SCIM on the other hand is a REST based technology

[1] http://www.simplecloud.info/

and thus, this thesis will also compare REST and SOAP in SOA based solutions. The goal is to support organizations in their decision making process on choosing whether to use REST.

The research question is divided to two sub-questions:

- What are the advantages and disadvantages of using REST compared to SOAP for SOA based solutions?
- Does SCIM provide needed operations and entity model representation for users and their access rights?

The first question is researched through a literary review on topics of REST, SOA and SOAP. There are three possible conclusions that can be drawn from the literary review:

- If no serious disadvantages are found, REST is suited to be used is SOA.
- If some serious disadvantages are found, but they are not critical, REST is suited to be used is SOA with considerations.
- If critical disadvantages are found, REST is not suited to be used in SOA.

The second research question is studied through designing three scenarios for on-premises IAM. The SCIM standard is reviewed and then the scenarios are modeled using SCIM. The topics for the scenarios are:

- Internal IAM: How to model user identity and access rights for a user?
- External IAM: How to model customer organizations and their users' identities in addition to the users of the service provider organization?
- Role restrictions: How to model restrictions to permissions given from a role?

For each scenario, information is provided on how it could be implemented with SCIM and how entity models could be designed. The potential restrictions related to each model are also presented. The models for implementing these scenarios are then tested in a proof-of-concept work. The work includes SCIM-service implementation based on the models and service consumers, which use SCIM-service for the user identity and access rights information. Success of the implementation is then evaluated according to following criteria:

- Do the models work and fulfill given requirements?
- Do the models require extensions to the SCIM standard?
- How complex is the service provider implementation for the model?
- How complex is the service consumer implementation for the model?

1.4 Structure of the Thesis

The chapter 2 discusses IAM and how it is used in SOA based architectures. The related IAM standards which are used in SOA are presented shortly. Then the three scenarios designed for the purposes of this study for the different use cases for on-premises IAM solutions are introduced.

The chapter 3 discusses the previous studies on IAM and SOA that are relevant to this study. Through the literary review, conclusions on advantages and disadvantages using REST services in SOA based solutions compared to SOAP services are presented. In the end of the chapter, a list of questions based on the review is provided to help organizations in their decision making process, highlighting some of the important matters.

In the chapter 4, the SCIM standard is looked at in detail. First the SCIM schema is discussed and also how the schema can be extended. Then the SCIM protocol for operations is discussed.

The chapter 5 focuses on how SCIM can be used in SOA. The work relies on the scenarios designed for this study and presented in chapter 2. The chapter includes a presentation of how the scenarios can be implemented with SCIM and what kind of entity model they would consist of.

The chapter 6 focuses on evaluating the presented models using a proof-of-concept work. The chapter ends in an outline of a process that organizations can use to assess if SCIM is suitable for their use. The process is similar to the one used in this thesis (details of the steps can be found in the appendix C). It starts from gathering requirements and then dividing and processing them in three parts.

The chapter 7 discusses future opportunities for studies on this subject. The last chapter presents the overall conclusions of the study on how well SCIM is suited to be used in SOA.

2. IDENTITY AND ACCESS MANAGEMENT

This chapter takes a closer look at IAM. The chapter starts with the evolution of IAM and then introduces different concepts and terms related to IAM. Then it is discussed how IAM is used in SOA. After that three different scenarios are presented. Based on the scenarios, requirements for IAM systems in SOA are listed.

2.1 Evolution of Identity and Access Management

The evolution of IAM can be divided into three waves in this thesis. The first wave of Identity Management (IdM) a was centralized IAM system for organization's internal users. The IAM system was responsible for handling the identity and access rights of the employees of an organization and the system helped integrating the information to other software systems in the organization.

The second wave of identity management started when business requirements included letting users of partner organizations or customers access information and services provided by the organization. For example, banks started to provide customers web access to services in order for them to be able to pay bills and access details of their accounts. This could be achieved by having a separate IAM system for external users or categorizing them as partner/external/customer users and adding them to the internal IAM system of the organization.

When customers or partners wanted to update their users information or access rights, they might have phoned or had some other procedure to notify the changes. The service desk of the organization then updated the modifications on behalf of the customer. The third step in IAM evolution came from outsourcing this administration of external users to customer and partner organizations. Outsourcing this procedure brought the organization cost savings and possibly also competitive advantage.

For this third step, providing user interface for maintaining users might be enough at the beginning. However, providing an Application Programming Interface (API) for customers would help integration to customer's own IAM system and processes. One possible downside in outsourcing user administration is security; the customer's employee responsible for the administration of the customer's users might not be trained well for the task, so it might lead to giving wrong permissions to users. Other risk is that opening access to IAM for external users might open new kinds of

Table 2.1: Three types to identify a user

Identification method	Example
something people hold	key
something people are	fingerprint
something people know	passphrase

attacks against the software systems of the service provider. The former risks can be mitigated by putting effort on the service user interface design and providing guides and training to the users.

2.2 Concepts and Terms

Identity in IAM means digital representation of a person, an organization or other legal entity. The digital identity is a collection of attributes about the entity. For a person they might be firstname, lastname, social security number and so on. Access in IAM comes from the term access rights and it means which resources user can access with which permissions. The term IdM is similar to term IAM and it means managing digital identities throughout their lifecycle.

IAM is closely related to acronym AAA, which means Authentication Authorization Auditing. Sometimes auditing is replaced with accounting, but the meaning is the same. *Authentication* is a term used for identifying a user who is challenged to provide identification to prove that the user is who she claims to be. There are three methods to prove identification: ”something people hold”, ”something people are” and ”something people know” (Linden 2009, p. 13). These are listed in table 2.1 with examples. Sometimes multiple methods are used for authentication in order to achieve stronger security.

Authorization is a term used for giving user access to protected resources. Before proceeding with authorization, user's identification must be verified, in other words the user must be authenticated. After that the access control procedure checks what roles (role-based access control (RBAC)) or attributes (attribute-based access control (ABAC)) the user has. If those roles or attributes satisfy the access rights requirements of the requested resource or operation, the user is granted access to it. Authorization can also use groups, in which groups are assigned to certain roles. Then the users that are members of the group are given the access rights which are assigned to the group.

Auditing has become an important part of the IAM system and IAM products usually have tools for producing audit reports. Auditing can involve checking the current state of the system (users and their permissions), systems logs, system's state in the past (e.g. what were the permissions of a specific user a year ago) and so on. The purpose of the auditing is two fold. First it aims to prevent problems before

they occur by regular inspections. Inspections usually provide reports of users, for example dangerous role combinations, users that should have their access rights revoked and so on. The second purpose is to provide accountability; enabling audit trail of the executed operations that can be traced to the real person(s). This must take into account changes to user access rights and to users' identity throughout the lifecycle of the identity. (Linden 2009, p. 17-19) So in case a user executes malicious operations, accountability enables identifying the person who executed the operations.

Identity *Federation* is a method for transferring identity and access rights information across organizations. As organizations usually have their own internal IAM-system, building a loose trust between the partner/customer organizations enables them to transfer an identity from one organization to the other. The organization which provides the identity is called Identity Provider (IdP) and the organization where the identity is transferred through federation is called Service Provider (SP). (Linden 2009, p. 52)

2.3 Identity Management in Service Oriented Architecture

IAM systems usually provide the following services in SOA to other systems; authentication, authorization and managing the lifecycle of a user identity (creating, reading, updating and deleting). All of the responsibilities do not have to be provided by a single service, they can be separated to multiple services.

Authentication and authorization standards exist for integrating the IAM-system with other systems, but for SOA based solutions there has not been a widely spread standardized API for service based IAM. The IAM softwares have lacked necessary plugins/components to provide identity and access management as a service. This has also been a problem in software as a service (SaaS) based solutions, where identities could not be automatically created for the service by a standard service API. Federation is one way to transfer identities to SaaS service, but it lacks methods to manage the whole lifecycle of an identity, for example a method for deleting identities.

Next, three scenarios for IAM in enterprise SOA are presented. These form the basis for evaluating SCIM's capabilities to act as a standardized API for service based IAM. The first scenario represents an organization, which has an IAM system for internal users. The second scenario represents an organization which has external and internal users. The third scenario extends the second one with fine-grained access rights requirements. Fine-grained access rights in this case means that authorization for a resource does not only depend on the user's roles.

The scenarios are based on a RBAC authorization model. Services define different access levels and depending which roles or attributes the user has, the services

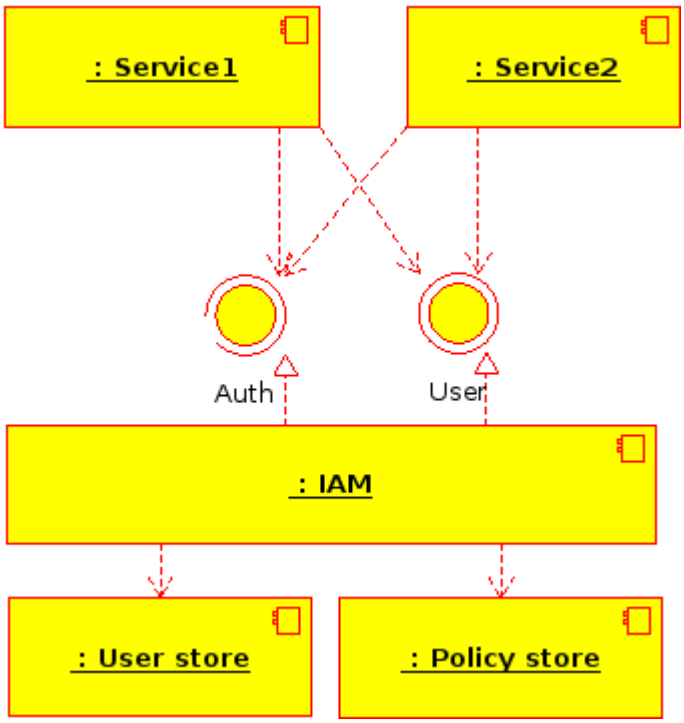

Figure 2.1: Architecture for IAM services in SOA

determine what the user can access in the service. An example service is presented in each scenario which uses the external IAM system for user identity and user's access rights information.

General architecture for these scenarios is shown in the figure 2.1. IAM service provides the authentication service (interface Auth in the figure) to other services and information about the users and their access rights (interface User in the figure). In the designed scenarios, the services only query data from the IAM system. Information in the IAM system, users and policies, is administrated in the IAM system.

2.3.1 Scenario 1: Service for Internal IAM

This scenario is the simplest and reflects the early stages of IAM evolution. In this scenario, the organization manages its own users and their access rights to organization's own services with an IAM software. The IAM software then provides this information as a service to the other applications and services of the organization. The requirements for the information schema are listed in the table 2.2. Functional requirements are listed in the table 2.3.

Table 2.2: Scenario 1 requirements

Requirement number	Description
101	Entity representation for user must exist.
102	Representation for a user must have attributes for the following: firstname(s), lastname, username.
103	Representation for a user must have attribute for user's email address.
104	Representation for a user must have attribute for user's phone number.
105	Representation for a user must have attribute for user's street address, postal code, city and country.
106	Representation for a user must have attribute for identification: social security number or employee number.
107	Representation for a user must have attribute to identify user's organizational department.
108	Users can be categorized to distinguish regular users from technical users. Technical users are non-human users.
109	Representation for access rights must support the RBAC model.
110	Representation for user's access rights, so a service can determine which roles the user has.
113	There must be status attribute for a user, to determine if the user is active (allowed to login) or deactivated.

Table 2.3: Scenario 1 functional requirements

Requirement number	Description
151	User and user's access rights information must be retrievable with one request from the SCIM service.

UML class diagram for this scenario is presented in the figure 2.2. The diagram expresses the model for user's access rights. For representing the identity of an employee there is an entity *User*. The user can have many access rights: the entity *Accessright* represents access rights. The *Accessright* provides information on what are the user's roles and groups.

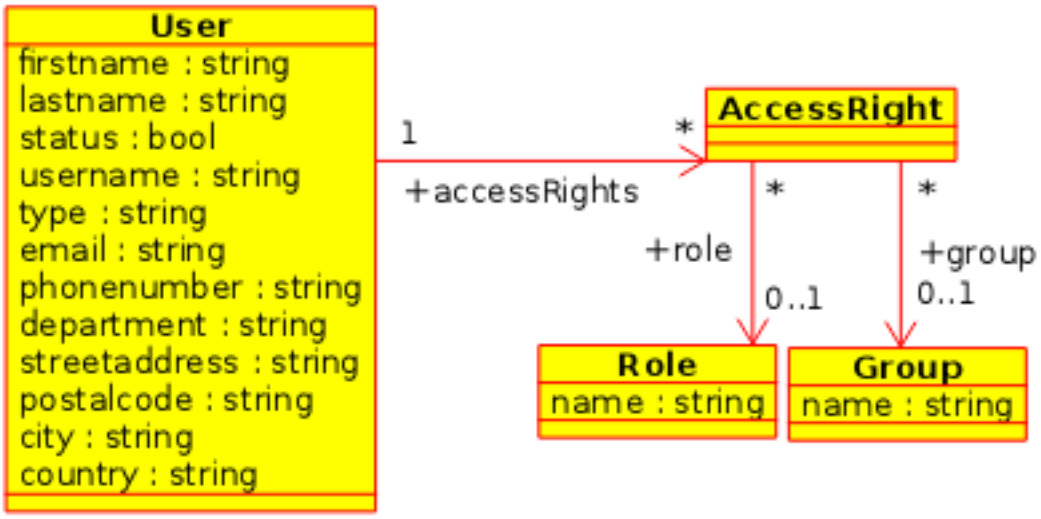

Figure 2.2: UML class diagram for the scenario 1 model

Here is an XML representation of what the employee's access rights representation might look like for this model:

```
<accessrights>
    <accessright>
        <role>
            <name>Employee</name>
        </role>
    </accessright>
    <accessright>
        <role>
            <name>Administrator</name>
        </role>
    </accessright>
</accessrights>
```

From the above XML it can be deduced that the user has the following roles: *Employee* and *Administrator*. The service using RBAC can decide what the user is entitled to access with those roles in the service.

2.3.2 Scenario 2: Service for External and Internal IAM

In this scenario the organization provides services to its users, like in the first scenario, but in addition it also provides services to its customers and partners. This means the model needs a way to represent organizations and links between organizations and users. A simple version would be adding an entity type *organization* and then linking the user to the organization. This has the downside that user can have link to only one organization: usually the organization is the user's employer.

However, in some business domains a user might have multiple links to different organizations, for example in the banking system a user might have access rights to his/her own bank account, employer's account, some other organizations' accounts related to hobbies and so on. To be better suited for these requirements, the model should provide multiple links between users and organizations.

In this case organization information must be included also in the access rights in addition to user-organization relationship; when a user has access to a service with a role, the information to which organization in the service the access right is linked to must be available. All the services might not need the organization information. The IAM service can leave it out and just return service-role pairs if it is not required. Services can also discard extra information, if it is not required to determine the user's access rights for the service.

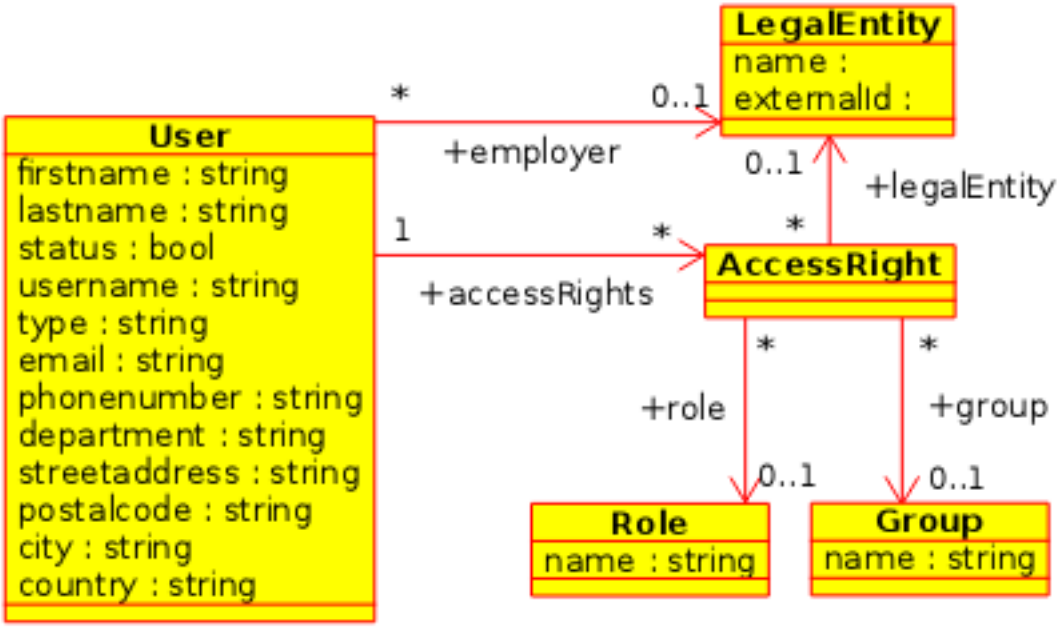

Figure 2.3: UML class diagram for the scenario 2 model

The class diagram is presented in the figure 2.3. There is a new entity, *LegalEntity*, which represents a person or an organization. Generally this entity represents customers and also the organization which is the service provider. The entity is linked to the users, so User-LegalEntity associations can be represented (for example employer-employee relationship). LegalEntity is also linked to the access rights, to indicate the LegalEntity, to which the access right gives permission. The consuming services can make authorization decisions in multi-tenancy setups. The LegalEntities should have a *name* attribute and some attribute for unique identification. The identification attribute can be a social security number, if the LegalEntity represents a real person, or a business id etc. In this study that attribute is named *externalid*.

Now an XML-representation of a user's access rights might be like the following:

```
<accessrights>
    <accessright>
        <role>
            <name>Customer</name>
        </role>
        <legalentity>
            <name>Organization1 Co</name>
            <externalid>342134-234</externalid>
        </legalentity>
    </accessright>
    <accessright>
        <role>
            <name>Customer</name>
        </role>
        <legalentity>
            <name>Company E Ltd</name>
            <externalid>2342235-213</externalid>
        </legalentity>
    </accessright>
    <accessright>
        <role>
            <name></name>
        </role>
    </accessright>
</accessrights>
```

It can be deduced that the user has three *accessrights*. The first two are for a service *crm* with roles *read*. They are restricted to LegalEntities Company E Ltd (externalid 2342235-213) and Organization1 Co (externalid 342134-234). Then the user has the role *admin* to the service *intranet*. This is not restricted to a certain LegalEntity. The table 2.4 has requirements for the IAM service for the scenario 2. The requirements from the scenario 1 also are valid for this scenario, see the tables 2.2 and 2.3.

Table 2.4: Scenario 2 requirements

Requirement number	Description
201	Entity representation for LegalEntity must exist. It must include attributes for name and externalId.
202	There must have a way to present association between the user and LegalEntity.
203	There must have a way to present association between access rights and LegalEntity.

2.3.3 Scenario 3: Service for External and Internal IAM with Access Rights Restrictions

This scenario extends the previous one. In this scenario, applications need to restrict role access to certain objects, meaning with a permission to a given role one can access only given objects. This requirement is useful for example in the banking industry, where a user might represent an organization, but might not have access to all accounts of the organization. Same kind of requirement is in the insurance industry, where a user might have access to some of the insurances of the organization, but not to all of them. The same logic can be followed as in the previous scenario when the access rights were restricted to certain legal entities. A new entity link to the accessrights is added, it is named *AccessRightEntitlement*. These entities are meant to represent for example a bank account or an insurance, so these entities have also an owner. For this, there is an association to LegalEntity to indicate the owner of the *AccessRightEntitlement*.

Other requirements from scenarios 1 and 2 apply to this scenario also. See the figure 2.4 the entity relationships of this scenario.

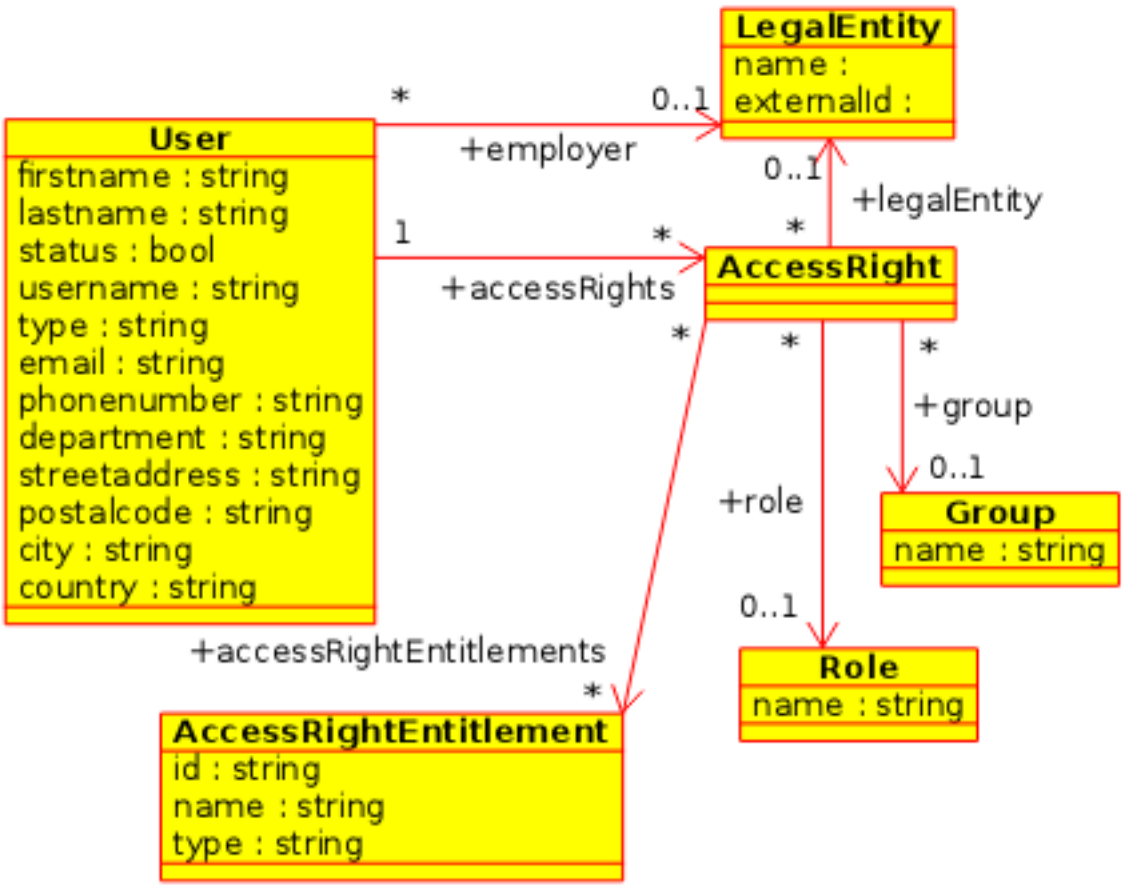

Figure 2.4: UML class diagram for the scenario 3 model

New requirements for this scenario are listed in table 2.5.

Table 2.5: Scenario 3 requirements

Requirement number	Description
301	Representation for AccessRightEntitlement must exist.
302	There must have a way to present association between LegalEntity and the AccessRightEntitlement.
303	Accessrights can be restricted to certain AccessRightEntitlements.

An example representation for a user's access rights in XML for this scenario is the following:

```
<accessrights>
    <accessright>
        <role>
            <name>Customer</name>
```

```
        </role>
        <legalentity>
            <name>Organization1 Co</name>
            <externalid>342134-234</externalid>
        </legalentity>
        <accessrightentitlements>
            <accessrightentitlement>
                <name>Account Organization1 Co</name>
                <id>1142112321-12321-2</externalid>
            </accessrightentitlement>
        </accessrightentitlements>
    </accessright>
    <accessright>
        <role>
            <name>Customer</name>
        </role>
        <legalentity>
            <name>Company E Ltd</name>
            <externalid>2342235-213</externalid>
        </legalentity>
        <accessrightentitlements>
            <accessrightentitlement>
               <name>Account Company E Ltd</name>
               <id>11428849-2434-21</externalid>
            </accessrightentitlement>
        </accessrightentitlements>
    </accessright>
    <accessright>
        <role>
            <name>CustomerAdmin</name>
        </role>
    </accessright>
</accessrights>
```

3. RELATED WORK

This chapter discusses previous studies that have been made on IAM as a service and on REST services in SOA based solutions. The literary review on the mentioned topics answers whether using REST in SOA is a viable alternative to using SOAP. In the end of the chapter, a list of questions is presented to help organizations' decision making process on whether to adapt REST technology.

3.1 Identity and Access Management

In this section, different architectures for implementing IAM services in SOA are presented based on previous studies. Two standards that are similar to SCIM are also discussed.

3.1.1 Identity and Access Management in Service Oriented Architecture

In the article "Identity as a Service – Towards a Service-Oriented Identity Management Architecture" Emig et al. (2007) discuss WSOA, where Identity and access management are provided as a service. The goal of the IdM architecture that the authors present is "to verify authorization for service usage at runtime by enabling access control" (Emig et al. 2007, p. 3). They propose a service-oriented identity management architecture, where access control consists of two parts: authentication (identifying the user) and authorization (checking if the user has access to resource). The two parts were designed to be provided as a service, the former was called *Security Token Service* and the latter *Policy Decision Point* (Emig et al. 2007, p. 4). The services use *Policy Store*, *Token Repository*, *User Directory* and *Service Registry* for processing requests (Emig et al. 2007, p. 3). Authors do not present detailed models of the policy store and the user directory.

This thesis uses a different approach than Emig et al. (2007). In the solution of this study, the IAM service provides user and access right information for the consuming services. This means that the services make the authorization decisions instead of delegating them to an external service. The models are also an important part of this study: what information is available on the users and how access rights are modeled.

Celesti et al. (2010) discuss Identity Management and problems in a federated Cloud Computing environment. They propose an InterCloud Identity Management Infrastructure (ICIMI) as a solution. It extends Identity Provider (IdP) /Service Provider (SP) model to the cloud computing environments and enables federated clouds to share resources with each other using trusted *IdentityProvides*. Their solution is implemented using Security Assertion Markup Language (SAML) profile. The models presented in this thesis are limited to on-premises solutions: the federated IdM is out of scope for this thesis.

3.1.2 Identity and Access Management Standards

There exists several standards related to IAM, for example different standards and protocols for authentication and authorization. For a more complete standard for IdM, in addition to SCIM, there seems to be only one good candidate, Service Provisioning Markup Language (SPML). Another similar but older standard is Lightweight Directory Access Protocol (LDAP) which is also discussed briefly in this section.

Service Provision Markup Language

SPML is an XML-based standard developed by Advancing Open Standards for the Information Security (OASIS) to provision and de-provision information about users, resources and services between and within organizations. Version 1 of the standard was approved in 2003 (Reed et al. 2003) and version two in 2006 (Bohren et al. 2006). It was said to be the standard to move identity in the cloud when the cloud computing and SaaS based services took the market. As an XML-based technology, it is very suited to be used within WSOA. However, the standard has not been widely implemented in IAM suites and in SaaS services and support for it has been nonexistent (Spencer 2012). This has been said to be due to complexities in the standard and has lead to very product-specific SPML implementations. One factor is that the SPML standard did not define entity types for users, services etc. The entity types had to be modeled separately by the implementing vendor. Work on a simpler standard for user identity and access rights provisioning started and later the SCIM standard was released. It seems the new emerging SCIM standard has gained more support and it will be more relevant in the future.

Lightweight Directory Access Protocol

Another popular older standard used in IAM is LDAP. It is an open protocol to exchange directory information across systems. It is widely used in directory servers,

such as Microsoft's Active Directory and OpenLDAP[1]. Many server applications have features to support retrieving identity and/or access rights for users using the LDAP protocol. There are multiple libraries and frameworks to help integration to different runtime environments. Usually LDAP servers are used as part of the organizations IAM solution, but LDAP based solutions usually lack necessary features for a complete IAM solution. Also the LDAP protocol uses ports 389 (and 636 for a secure connection) to which traffic might not pass through different firewalls. This hinders its usage on off-premises solutions.

3.2 REST Services in Service-Oriented Architecture

Comparing REST and SOAP based solutions for using in SOA has been the subject of many studies. In order to determine the advantages and disadvantages of using REST in SOA, some of the studies are reviewed next.

3.2.1 REST vs SOAP

Pautasso et al. (2008) compare SOAP and REST in SOA in their article "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision". They look at how many architectual decisions must be made when implementing services with both styles. According to the comparison both styles are similar at the fundamental level. Both can be used to build services but the whole SOAP stack with different WS-* standards is more mature[2]. The authors recommended that REST should be used for more ad hoc mashup type services and SOAP / WS-* stack should be used in the enterprise application integrations.

In the article "SOA-Readiness of REST", authors Gorski et al. (2014) study how mature REST services are and whether they are ready to be used in SOA. They present two major problems: the first one is the lack of service discoverability and service registry attributes in REST. The authors conclude that standardization on those aspects is required for REST. The second problem is how different REST frameworks consume request and produce responses differently. This hinders usage in heterogeneous environments and requires skilled developers to get around the differences the frameworks produce. (Gorski et al. 2014)

The articles were published six years apart. It seems that between the publication of the articles, the tools and the frameworks of REST have evolved. However, based on the studies SOAP still seems more mature for the enterprise-level.

One advantage REST has is that it is independent of data format (Fielding 2000, p. 90-92). It can be used with for example JavaSript Object Notation (JSON) or

[1]http://www.openldap.org/

[2]WS-* refers to the many standards in SOAP of which most begin with WS-

XML. This allows choosing suitable data format for different use cases and if/when better data formats emerge in the future, REST services can adapt them. SOAP on the other hand is tied to XML.

3.2.2 Standards

Pautasso et al. (2008) and Gorski et al. (2014) raise the lack of standards as a problem for REST compared to SOAP. SOAP is often referred as WS-* stack for the many formal extensions[3][4] that have been defined for it. Examples of SOAP related standards and extensions include:

- WS-Addressing, extension to add routing data to a SOAP message
- WS-Discovery, extension to locate services on a local network
- WS-Federation, extension to federate identity between security realms
- WS-Policy, extension to allow service advertise its policy (security, quality, etc) to consumers
- WS-Security, extension to provide end-to-end secure messaging
- WS-Trust, extension to WS-Security to use security tokens and establish trust relations

Although there might not be formal standards in REST, it does not mean that similar kind of feature could not be added. Peng et al. (2009) propose authentication system for REST based on the WS-security UsernameToken attributes that would be added to HTTP headers. Serme et al. (2012) in the article "Enabling Message Security for RESTful Services" implement message level security for REST services. The implementation included signing and encrypting messages for RESTfull services. It uses HTTP headers to add necessary attributes describing the signature and encryption metadata for the messages. The implementation tries to add similar features that the WS-Security provides, but respecting the RESTfull philosophy and minimizing processing overhead for the service consumers.

The presented extensions above are using the HTTP headers for the extension attributes and so they do not have to alter the message or the URI parameters. This reduces processing overhead for the service consumers (Serme et al. 2012, p. 2). Tight integration with HTTP enables REST to use HTTP's features like HTTP headers. The SOAP counterparts on the other hand add the attributes to the XML message. Other similar example is the status codes for error handling: REST can

[3]https://www.oasis-open.org/standards
[4]https://www.w3.org/Submission/

send error message along with setting proper HTTP error code. Furthermore, SOAP was designed to support also other transport methods than HTTP, although HTTP is the most common use case. For this, SOAP has its own Soap Fault XML elements to indicate error in processing request in addition to HTTP status codes (Gudgin et al. 2007).

These examples show that that these kind of extensions can be added for REST services. However, these extensions are not (de-facto) standards (at least not yet) and may lead to different implementations and then to difficulties with interoperability.

3.2.3 Enterprise Service Bus

In their paper "A Multi-Language Service-Oriented Architecture Using an Enterprise Service Bus" authors Sward and Whitacre (2008) explore the use of ESB with both SOAP service and REST service. These services were programmed in two different languages, Java and ADA. They used open source ESB, called Mule[5], which has specific components for integrating REST based services, to produce and consume REST services in addition to SOAP services. In their experimentation it is easy to support both SOAP and REST services with ESB if the ESB software has support for both. (Sward and Whitacre 2008)

3.2.4 Performance

In a study by Aihkisalo and Paaso (2012), the authors measure the performance between REST and SOAP based services. The tests measured round trip time from client's request to the server's response. They also included measurements from inside the service implementation, which gives data on how much time was spent in different tasks during the service's request-response processing. The test client and service ran on the same computer, eliminating network latency. Implementations used the same business logic, only the interface to the services was varied. Implementations included different transmission data formats that were: REST-XML, REST-JSON, REST-GPB (Google Protocol Buffers)[6], SOAP-XML and SOAP-XOP/MTOM (XML-binary Optimized Packaging[7]/Message Transmission Optimization Mechanism[8]). In the results, the REST-XML and REST-JSON based implementations had 4-5 fold smaller round trip time than SOAP counterpart. With the fastest REST-GPB implementation the difference was 5-10 times. The measurements from the REST-XML implementation show that XML was not the sole reason for SOAP's higher latency in the tests. It was the complexity of

[5]https://www.mulesoft.com/
[6]https://developers.google.com/protocol-buffers/
[7]https://www.w3.org/TR/xop10/
[8]https://www.w3.org/TR/soap12-mtom/

SOAP. Though using JSON instead of XML gives also a performance boost when transferring data through network because it makes the message size smaller. In their tests, this did not have an effect because the network latency was minimal. They conclude that the REST services had better performance in all the test cases and they attribute this to the complexities of SOAP.

In the article "Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration", Kumari and Rath (2015) arrive at the same conclusion. They implemented LoanBroker-service in ESB based architecture and monitored throughput and response time for the performance evaluation. In their test setup, services were accessed over network and the network latency increased response time on both service implementations. This decreased the request processing of the services percentagewise and the performance difference was not as drastic as in the study by Aihkisalo and Paaso (2012). Kumari and Rath (2015) found out that REST based services have better performance on the measured throughout and response time than the SOAP counterpart. They attribute this to the heaviness of XML-based SOAP. When they send larger loan files, the XML-based markup is smaller in terms of percentage. In this case, the difference was not as big as in other tests, but still the REST based service had advantage over the SOAP based service.

3.2.5 Is REST ready?

Based on the reviewed articles, it can be concluded that REST is suitable for implementing services for SOA based solutions. Pautasso et al. (2008) and Gorski et al. (2014) raise alerts that REST lacks good tooling and standardization. This is related to the fact that REST is based more on conventions than actual standards. The SOAP world has more standards and it has been the industrial de-facto solution in the 2000s. There exists a lot of tools and frameworks for different languages and runtime environments and it is convenient to build services with SOAP (Kumari and Rath 2015). Since REST is an emerging technology, an assumption can be made that better tools and frameworks will be available for REST in the future. This is a serious but not critical issue. For implementing on-premises solutions it is recommended to have internal implementation standards in place and experienced developers to make sure the services adhere to the standard and use same conventions between different services. This is more work than designing architecture and guidelines to use relevant WS-* stack standards.

Implementing services with REST brings also benefits. REST is independent of data format, so services can be implemented using a data format suitable for the particular service. Another benefit is performance. Two articles compared the performance of REST and SOAP based services and both conclude that REST based implementation has better performance. This is worth noting if the services are going

to be under heavy load or need fast responses. Also tighter integration with HTTP enables to use HTTP's features where SOAP implements its own XML elements. For example adding digital signature for messages can be achieved in REST by setting HTTP header attributes, SOAP on the other hand encodes this information to the XML message.

This review did not find any single critical issue that would prevent REST usage in SOA based solutions. However, every organization has to evaluate the benefits and drawbacks in their particular use case. An organization that has already invested in SOAP technology can use the following questions in its decision making process on whether to use REST:

- If the organization has an ESB, does it also support REST based services?
- Can REST based services easily integrate to other mandatory services the organization has, for example authentication service or monitoring service?
- What are the security requirements for services in the organization, can REST based services fulfill security requirements like message security?
- Would services benefit from using other data formats instead of XML?
- How REST would fit in the software architecture of the organization?
- Do the services have strict performance requirements in which using REST would be beneficial?
- Do the tools the organization uses, like software testing tools, support REST?
- Are the developers familiar with REST or is the organization ready to invest in training?

4. SYSTEM FOR CROSS-DOMAIN IDENTITY MANAGEMENT

SCIM is a protocol for managing user identities between different domains (Mortimore et al. 2012). It is designed to transfer identity information from the IAM system of an organization to the cloud services the organization uses. The SCIM standard consists of two parts: the core schema standard provides schemas to describe resources (Hunt et al. 2015b) and the REST API protocol defines CRUD operations (create, read, update, delete) for the resources (Hunt et al. 2015a).

Version 1.1. of the SCIM schema descriptions (Mortimore et al. 2012) and protocol definition (Drake et al. 2012) was released in July 2012. In September 2015 the version 2.0 was released. In this thesis the version 2.0 is used if not otherwise mentioned.

This chapter discusses the SCIM standard in detail. First there is a short introduction and then discussion on what kind of entity model the SCIM schema has. After that the SCIM protocol is looked at. The details about SCIM discussed in this chapter are used to model the scenarios presented earlier and to implement SCIM service in the proof-of-concept.

4.1 Schema

The SCIM schema consists of three core entity types that are called resource types. These are *User*, *Group* and *Enterprise User*. The last one is an extension to the *User* resource type. In addition to the core schema resources, there is a schema *Service Provider Configuration* for service providers to describe their service and *ResourceType* schema for providing metadata information about the resource types. In the *Service Provider Configuration* service, the provider can describe for example which query features they support and what kind of authentication mechanism is required to access the service.

The SCIM standard lists eight primitive data types for defining resource schemas; String, Boolean, Decimal, Integer, Datetime, Binary, Reference and Complex. These data types are used in schema attributes as basis for resource types and they are derived from JSON datatypes[1]. The Reference-type is used to reference other re-

[1]RFC7159 https://tools.ietf.org/html/rfc7159

sources and complex attribute type represents JSON object, which means it can have its own sub-attributes. Attributes that can contain multiple values are marked as *Multi-Valued Attributes*. Otherwise attributes are called *Singular Attributes*. The default type for attributes is *String*. Most of the attributes are optional and can be left out, the required attributes are noted in the standard. (Hunt et al. 2015b, p. 8-10)

The core resources *User* and *Group* types have common mandatory attributes; these are id, externalId and meta. The meta is a complex attribute containing the following attributes: resourceType, created, lastModified, location and version. The attribute version is optional and can have the same content as ServiceProvider's standard versioning of entities.

In addition to common attributes, each resource type has core attributes. If the resource type is a schema extension to another resource type, it has also extended attributes. The resource representation of a resource type must contain *schemas* attribute which lists namespaces for the resource's schema attributes. If the mandatory common attributes for resource types are defined as a class Common attributes, the SCIM resources can be presented as class diagram (see figure 4.1). Some of the attributes are left out in order not to clutter the image.

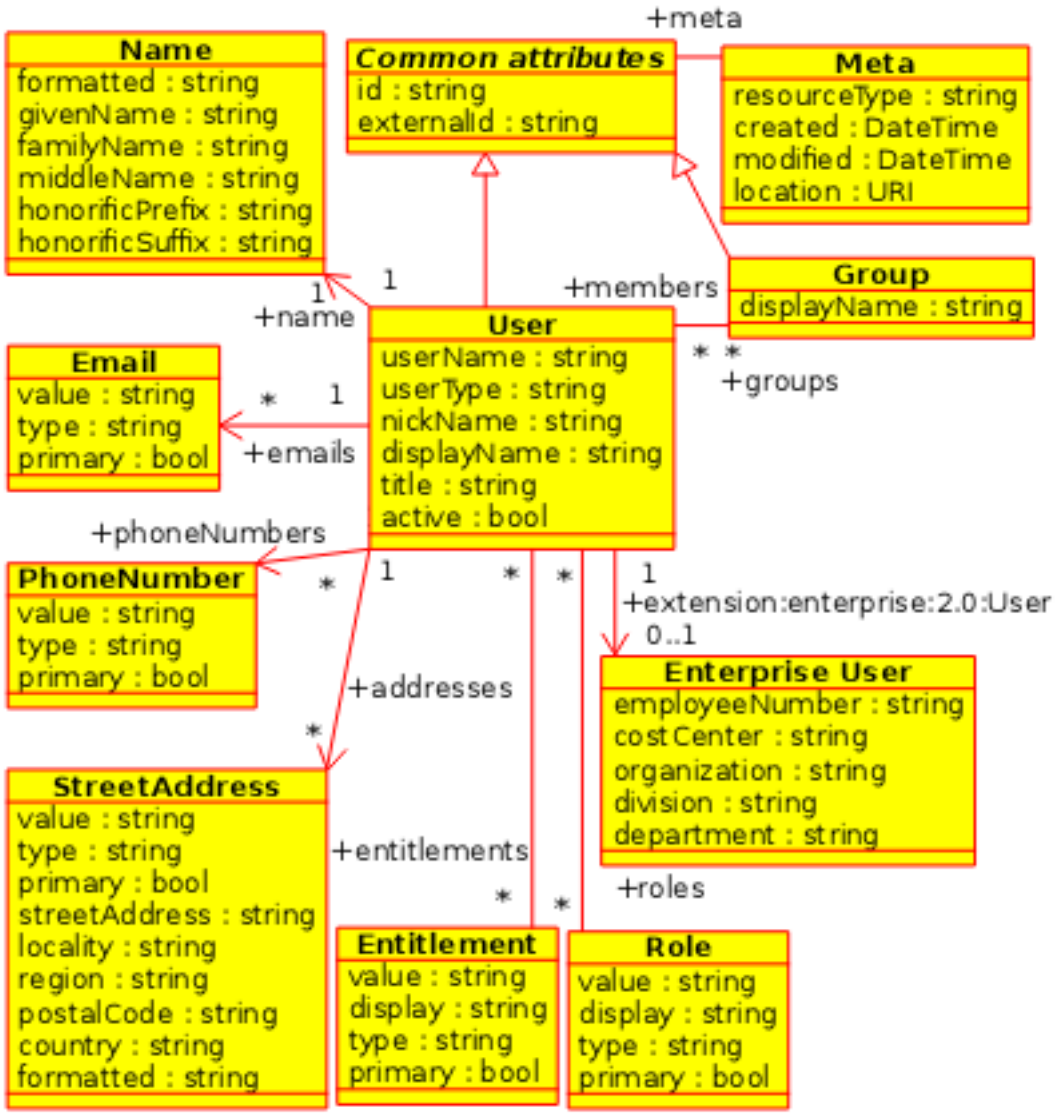

Figure 4.1: SCIM UML class diagram

4.1.1 User Schema

User schema represents information about users (see the listing 4.1 for minimal example representation of a user). It has attributes for describing user's name, addresses, phone number details, email addresses and so on. It has schema identification URI *urn:ietf:params:scim:schemas:core:2.0:User*. For representing users access rights there are three different attributes: groups, entitlements and roles. Groups attribute list the groups that the user is a member of. The description for *group* resource is: "The values are meant to enable expression of common group-based or role-based access control models, although no explicit authorization model is defined." (Hunt et al. 2015b, p 18-23). Entitlements is a list of entitlements that the user has. These may be additional access rights to objects or services. There is no

vocabulary or schema defined. Roles is a list of roles, which describe who the user is. SCIM standard has the following examples: *Student* and *Faculty* (Hunt et al. 2015b, p 18-24). Linden (2009) explains roles as "attributes describing the user's relationship to the organisation in order to facilitate access control". This is a good description of roles in SCIM. No vocabulary or syntax is provided for roles in the SCIM standard.

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:User"]
    ,
  "id": "2819c223-7f76-453a-919d-413861904646",
  "userName": "bjensen@example.com",
  "meta": {
    "resourceType": "User",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff590\"",
    "location":
    "https://example.com/v2/Users/2819c223-7f76-453a-919d
      -413861904646"
  }
}
```

Listing 4.1: Minimal SCIM user representation (Hunt et al. 2015b, p. 33)

4.1.2 Group Schema

The Group schema represents groups (see the listing 4.2 for example representation of a group). It has schema identification URI *urn:ietf: params:scim:schemas:core:2.0: Group*. In addition to the common attributes it has two additional attributes: displayName and members. DisplayName is the displayed name of the group and members is a list of references to users. These users are the members of the group. It is good to notice that the displayName is not a unique attribute: to identity groups from each other the group's id attribute must be used.

Groups are meant to represent common group-based or role-based access control models. A group can contain other groups. Members of a group containing other groups also get permissions from all the sub groups.

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Group"
    ],
```

```
  "id": "e9e30dba-f08f-4109-8486-d5c6a331660a",
  "displayName": "Tour Guides",
  "members": [
  {
    "value": "2819c223-7f76-453a-919d-413861904646",
    "$ref":
    "https://example.com/v2/Users/2819c223-7f76-453a-919d
       -413861904646",
    "display": "Babs Jensen"
  },
  {
    "value": "902c246b-6245-4190-8e05-00816be7344a",
    "$ref":
    "https://example.com/v2/Users/902c246b-6245-4190-8e05
       -00816be7344a",
    "display": "Mandy Pepperidge"
  }
  ],
  "meta": {
    "resourceType": "Group",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff592\"",
    "location":
    "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-
       d5c6a331660a"
  }
}
```

Listing 4.2: SCIM group representation (Hunt et al. 2015b, p. 42)

4.1.3 Enterprise User Schema Extension

Enterprise User Schema Extension extends the user schema to represent users which belong to or act on behalf of organizations or enterprises. It has the following identification schema URI *urn:ietf:params:scim:schemas:extension:enterprise:2.0:User*. The schema adds the following singular attributes: employeeNumber, costCenter, organization, division, department and manager. The manager is a complex attribute to reference the user's manager. It has the following attributes: value, $ref and displayName. (Hunt et al. 2015b, p 25-26)

4.1.4 Service Provider Configuration Schema

The service provider configuration schema has only read-only attributes. These attributes define if the feature the attribute refers to is supported by the service. The schema has the following identification schema URI *urn:ietf:params:scim:schemas:core:2.0:ServiceProviderConfig*. The schema has one singular string attribute, documentationUri, and the following singular complex attributes: patch, bulk, filter, changePassword, sort and etag. The schema also has one multi-value complex attribute called authenticationSchemes, which lists different authentication methods that can be used to access the service. It has the following attributes: name, description, specUri, documentationUri and type. The name and description are string attributes and specUri and documentationUri are HTTP URL addressable attributes. The attribute type can have the following values: oauthbearertoken, oauth, oauth2, httpbasic and httpdigest.

4.1.5 Extending the SCIM Schema

There are two ways to extend SCIM to the needs of the service provider: one way is to define new resources and the second is to extend the core resource types. Extensions allow to add more attributes to available resource types. The extended attributes are added to their own namespace, so they are distinguished from other attributes. The extended attributes should avoid redefining any attributes from the SCIM standard. (Hunt et al. 2015b, p. 17)

In the SCIM version 2, SCIM allows to define new resource types (Hunt et al. 2015b). This adds more capabilities for extending SCIM to wide variety use cases. Providing new resource types requires defining new resource type using *ResourceType* schema and define the resource type's schema. The schema definitions has a schema URI *urn:ietf:params:scim:schemas:core:2.0:Schema*. The definition has four attributes: schema type, id, name and description. Then there is a complex attribute called attributes which describe all the attributes of the schema, their type, sub-attributes (an attribute with sub-attributes is referred as a complex attribute), mutability and so on. The SCIM standard imposes one restriction; a complex-attribute must not contain another complex-attribute as a sub-attribute. (Hunt et al. 2015b, p 29-32)

4.1.6 User's Access Rights

The SCIM schema does not impose any particular schema for users' access rights, but the *Group* resource is meant to be used in RBAC or group based access right models. In addition to the groups, there are attributes *entitlements* and *roles* in the *User* schema, which can be used to encode access rights. Both *entitlements* and *roles* have

the same sub-attributes: value (data type string), display (data type string), type (data type string) and primary (data type boolean). Similar access rights models can be constructed using different attributes. The access right attributes are used in the following context:

- Role: the something user is, represents user's relationship to the organization, for example "Student"
- Group: something the user belongs to in the organization, department, teams, projects and so on. For example Administrators.
- Entitlement: something the user has, additional access right. One example is a bank account

For RBAC based authorization models, the SCIM schema is sufficient. However, in multi-tenancy setups (multi-tenancy will be defined and discussed in detail later on), access rights do not have explicit definition on which organization access rights are valid for. User's attributes have an attribute to dictate user's organization, but it is a singular attribute so a user can not have different access rights for different organizations. This requires extending the SCIM schema.

4.2 Protocol

The SCIM protocol defines operations to access and modify SCIM resources. First there is discussion about the operations and then on other service related information: authentication, authorization and multi-tenancy.

4.2.1 Operations

The SCIM protocol is based on HTTP and follows the REST style specification of CRUD (create, read, update, delete) operations for the resources. JSON is used to send and receive data to and from the service in the http message body. SCIM uses media type *application/scim+json*. Table 4.1 lists the HTTP methods the SCIM uses and a description on what the method is used for. Table 4.2 lists resource endpoints and which http methods the resource endpoint supports. Some methods are optional, for example service provider might leave support for PATCH method out from User-resource endpoint.

Table 4.1: SCIM HTTP Methods (Hunt et al. 2015a, p. 8)

HTTP method	SCIM Usage
GET	Retrieves one or more complete or partial resources.
POST	Depending on the endpoint, creates new resources, creates a search request, or MAY be used to bulk-modify resources.
PUT	Modifies a resource by replacing existing attributes with a specified set of replacement attributes (replace). PUT MUST NOT be used to create new resources.
PATCH	Modifies a resource with a set of client-specified changes (partial update).
DELETE	Deletes a resource.

Table 4.2: SCIM HTTP Methods (Hunt et al. 2015a, p. 9)

Resource	Endpoint	Operations	Description
User	/Users	GET, POST, PUT, PATCH, DELETE	Retrieve, add, modify Users.
Group	/Groups	GET, POST, PUT, PATCH, DELETE	Retrieve, add, modify Groups.
Self	/Me	GET, POST, PUT, PATCH, DELETE	Alias for operations against a resource mapped to an authenticated subject (e.g., User).
Service provider config	/ServiceProviderConfig	GET	Retrieve service provider's configuration.
Resource type	/ResourceTypes	GET	Retrieve supported resource types.
Schema	/Schemas	GET	Retrieve one or more supported schemas.
Bulk	/Bulk	POST	Bulk updates to one or more resources resources at the same request, instead of the default update for one resource per request.
Search	[prefix]/.search	POST	Search from system root or within a resource endpoint for one or more resource types using POST.

There are two ways to retrieve users from the SCIM service: query them or access them using the resource id. Both of these methods use HTTP method GET (see the table 4.2 for list of the endpoints for resources and which operations they accept). Schema extensions add attributes to own namespace which are then returned from the resource endpoint which it extends (*Enterprise User* does not have a separate

endpoint, it uses the same endpoint as the *User* resource). Querying requires sending the query parameters as a URL parameter for the endpoint. The protocol defines optional features for querying, like filtering, pagination and sorting.

4.2.2 Authentication and Authorization for Service

SCIM does not define a mechanism for authenticating and authorizing users, this is left for service providers to decide and implement. The protocol specification lists different options for authenticating consumers to the SCIM service. These include TLS Client Authentication, HOBA Authentication, Bearer Tokens, PoP Tokens or Cookies. (Hunt et al. 2015a, p. 4-5) The use of Basic Authentication is not recommended. Though no authentication method is defined, service provider must be able to map authenticated user to access control policy to determine user's access rights.

4.2.3 Multi-tenancy

A SCIM service provider might expose the service to multiple different customers and the customers may access resources that are shared between different customers. This kind of setup is called multi-tenancy. SCIM does not itself define a scheme for multi-tenancy. It is an optional feature, but the protocol has a separate chapter for multi-tenancy setups. It lists the following cases (Hunt et al. 2015a, p. 75-77):

1. "All clients share all resources (no tenancy)."
2. "Each single client creates and accesses a private subset of resources (1 client:1 Tenant)."
3. "Sets of clients share sets of resources (M clients:1 Tenant)."
4. "One client can create and access several private subsets of resources (1 client:M Tenants)."

For an organization's on-premises solution for internal use, the first option "All clients share all resources (no tenancy)" is used. If multi-tenancy is required by the scenarios defined in chapter 2, it can be implemented by extending the SCIM schema.

5. USING SCIM IN SERVICE-ORIENTED ARCHITECTURE

This chapter focuses on using SCIM in SOA and answers the second research question presented in the chapter 1: Does SCIM provide required operations and representation for users and their access rights? The three scenarios presented in the chapter 2 are modeled using the SCIM schema to measure the capabilities of SCIM. For each of the scenarios, it is shown how the model fulfills the requirements and how SCIM can be used for different kind of use cases.

5.1 Architecture and Usage

A SOA-style architecture for an organization's IAM system with SCIM interface is presented in the figure 5.1. An underlying assumption is that the organization has a set of services and an IAM system. The IAM system provides an authentication service and a SCIM service for other services. The SCIM service can also be a separate adapter-style component if the organization's IAM software does not support SCIM API.

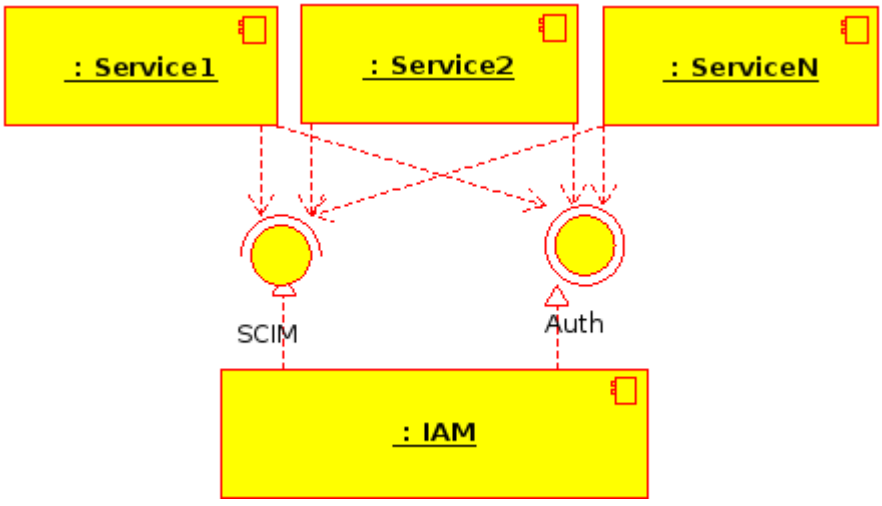

Figure 5.1: Architecture when using SCIM with IAM system

The SCIM service provides identity and access rights information for other services. The data the SCIM service uses and returns is administered in the IAM system. This simplifies implementation for the SCIM service as it is only used to

query data. If an organization decides to open its SCIM service for its customers for them to use it (add users and change their access rights), SCIM service must support adding, modifying and deleting users. Also, if the organization has some other requirement that needs the above mentioned functionality, then it must be implemented. However, if only the IAM system needs to be used to administer the data, then the SCIM service only needs to support retrieving resources.

5.1.1 Authorization

Authorization decisions are made in the consuming services. As noted in the chapter 4, the SCIM standard does not give an authorization model for the SCIM service. For this thesis, a simple authorization model for the SCIM service is presented in the table 5.1. Regular active users can access their own information, technical users (userType attribute has the value *ServiceAccount*) can access every resource. Non-authenticated users or de-activated users can not access any resource.

Table 5.1: Access levels for the SCIM service

User type	Access to	Requirement
non-authenticated	-	-
regular user	read access to own information	active attribute is true, userType attribute is *User*
technical user	read access to every resource	active attribute is true, userType attribute is *ServiceAccount*

5.2 Scenario Models

Next, the details on how all the three scenarios can be modeled with SCIM are discussed. In addition to the models, example algorithms are presented for each scenario. The algorithms can be used to check if a user has the required roles and group memberships to access a resource. At the end of each scenario, an example service is presented as a demonstration on how the model of the scenario can be used.

5.2.1 Scenario 1

To model the scenario 1, the requirements are discussed in three parts: first the requirements for SCIM User schema are addressed, then the requirements for access rights and at last the functional requirements.

User Attributes

For representing User entity SCIM has the User resource. As seen in the chapter 4, SCIM has rich a set of attributes for User. The table 5.2 maps the scenario 1 requirements for the SCIM User and Enterprise User Schema. SCIM provides multi-value attributes for emails, phonenumbers and addresses, so a user can have different email addresses, phone numbers and street addresses for work and home. This was not required, but it is an extra benefit. The User schema does not have attributes to state user's information, like department and employee id. This requires using the Enterprise User extension. Together they fulfill the requirements for the user's attributes.

Table 5.2: Scenario 1, 2 and 3 user attribute mappings

Requirement number	Attribute description	SCIM attribute name
102	firstname	User Schema name attribute's sub-attribute givenName
102	lastname	User Schema name attribute's sub-attribute familyName
102	username	User Schema attribute userName
103	email	User Schema attribute emails
104	phonenumber	User Schema attribute phoneNumbers
105	street address	User Schema attribute addresses
106	identification	Enterprise User Schema attribute employeeNumber
107	department information	Enterprise User Schema attribute department
108	user type	User Schema attribute userType
113	user status	User Schema attribute active

Access Rights

In the SCIM standard, groups are used to represent role-based or group-based access models (Hunt et al. 2015b, p. 24). Also the roles attribute can be used to describe user's roles. Combination of them is enough to address basic role- and group-based authorization requirements which the scenario 1 has: a user can have certain roles and can be a member of certain groups.

Functional Requirements

This scenario has one functional attribute, requirement number 151 (see the table 2.3). It states that only one request to the SCIM service should be needed in order to retrieve information on a specific user. The SCIM protocol defines usage for HTTP method GET to retrieve user's resource by its URI. The resource's URI has UUID for identifying the user, so authentication service must return UUID information or the full URI user's resource so other services can retrieve the user information from the SCIM service. Consuming services can check from the SCIM service's response, if a user has a required role and group to access a resource. The algorithm below is an example to check if a user has the required role and group membership to access a certain resource. The required role and group id is given as a parameter. Appendix B has a response from the SCIM service's *User* endpoint for the scenario 1.

Algorithm 1 Algorithm to check if a user has a certain role and certain group membership

```
function employeeUserAuthz(roleName, groupId, user)
    roles ← user[roles]
    groups ← user[groups]
    for all r in roles do                          ▷ Check if user has a certain role
        if r[value] = roleName then
            for all g in groups do    ▷ Check if user is a member of a given group
                if g[value] = groupId then
                    return true
                end if
            end for
            return false
        end if
    end for
    return false
end function
```

Example Service: Intranet

The example service for the scenario 1 is called *intranet*. It is a wiki-like knowledge base application for the organization. It consists of different namespaces, to which employees can add content. The content can be wiki pages, documents and so on. Access to the namespaces can be restricted to require certain role or group membership. For example members of a team or a department inside the organization belong to the same group and that group could have a namespace in the application to share content within the team.

This example service consists of four namespaces: *employees*, *organization*, *administrators* and *sales*. The namespaces *employee* and *organization* are for all employees

of the organization, so all active users (attribute active value set to true) with role *Employee* can access the namespaces. The other namespaces require group membership in addition to the *Employee* role. The sales namespace is for the members of a sales team and the administrators namespace is for members of the administrators team. Members of the given group have access to the group's namespace in the service. The table 5.3 shows the different access levels to the intranet service. The example algorithm presented earlier can be used to check that user has required role and group membership to a namespace.

Table 5.3: Intranet service's access levels

Access level	role name	description
General access	Employee	Access general employee knowledge base.
Specific group namespaces	Employee	Group membership gives access to specific namespaces in the intranet, for example administrators group membership gives access to administrators namespace.

5.2.2 Scenario 2

The scenario 2 has no new functional requirements, but it has new requirements for user attributes and access rights. The requirements in this scenario need an entity to represent customers. The requirements refer this entity as *LegalEntity*. It must have associations to User (for employee association) and to access rights (for associating the customer to the access rights). The Enterprise User Schema in SCIM does have attributes for department or organization's name, but no unique identifying attribute for the organization, like *externalId* or *organizationid*. For this requirement, the scenario 2 adds new extension for SCIM User schema to include information about employer of the customer user. The extension is called *Customer User* and it includes three attributes to point the employer: employerName, employerExternalId and employerType. These satisfy the requirement numbers 201 and 202.

Access Rights

Information about *Legal Entity* mentioned above must also be linked to the access rights. As the roles-attribute in SCIM cannot be extended, another attribute is added to the previous extension schema to express access rights the user has to different customers (satisfies the requirement number 203). This attribute is called customerroles. It includes information about the customer (legalEntityExternalId,

legalEntityName and legalEntityType) and a list of roles the user has for the customer.

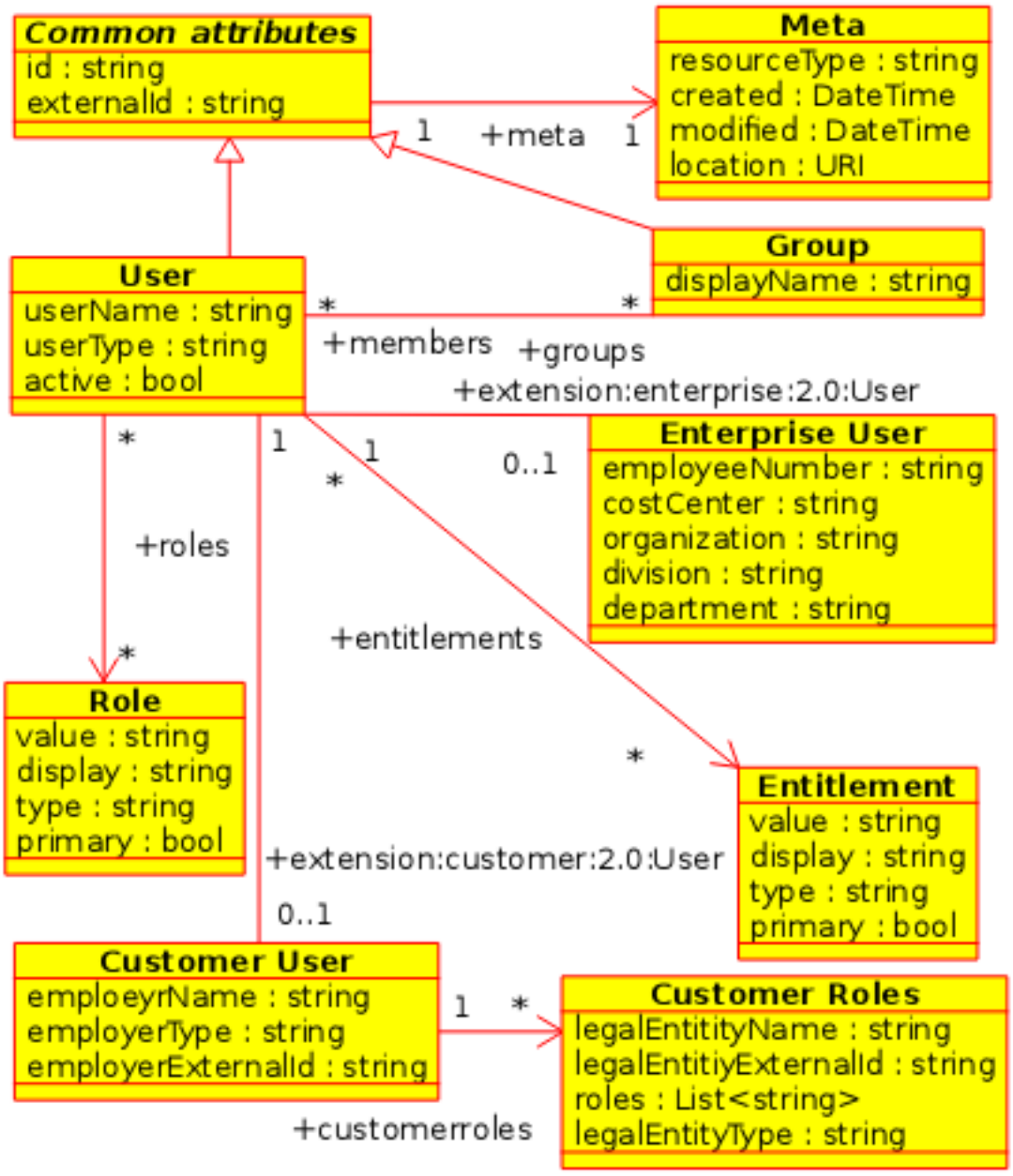

Figure 5.2: SCIM Scenario 2 UML class diagram for access rights

The SCIM model for the scenario 2 is seen in the figure 5.2. The SCIM User schema extension outline for representing access rights to customers and user's employer connection is below. For the whole extension definition, see the appendix A.

```
{
 "id" : "urn:korkeala:params:scim:schemas:extension:
    customer2.0:User",
```

```
"name" : "Customer User",
"attributes" : [
  {
    "name" : "employerName",
    "type" : "string",
    "multiValued" : false,
  },
  {
    "name" : "employerExternalId",
    "type" : "string",
    "multiValued" : false,
  },
  {
    "name" : "employerType",
    "type" : "string",
    "multiValued" : false,
  },
  {
  "name" : "customerroles",
  "type" : "complex",
  "multiValued" : true,
  "required" : false,
  "subAttributes" : [
  {
    "name" : "legalEntityExternalId",
    "type" : "string",
    "multiValued" : false
  },
  {
    "name" : "legalEntityName",
    "type" : "string",
    "multiValued" : false
  },
  {
    "name" : "legalEntityType",
    "type" : "string",
    "multiValued" : false
  },
  {
```

```
      "name" : "roles",
      "type" : "string",
      "multiValued" : true
    }
    ]
    }
 ]
}
```

Listing 5.1: "Outline for the customer user extension to User schema"

In this scenario, there are two cases that must be checked to define whether a user has the required access rights to a resource: if the user is a customer user or if the user is an employee of the organization. Below is an example algorithm for checking access rights of a user (see the algorithm 2). In the former case (the function *customerUserAuthz* in the algorithm 2), it checks that the user has a required access role and a certain *Legal Entity* in the same customerroles definition and that the user has a role that identifies it as a customer user. The access role, the role to identify user as a customer user and the externalid for *Legal Entity* are given as parameters. For the second case refer to the algorithm 1 from the scenario 1. It checks that the user has a required role and group membership to access a resource. See the appendix B for an example response from the SCIM service for a user which has customerroles.

Example Service: CRM

The example service for the scenario 2 is called *crm*. It is a customer-relationship-management service and it is intended to be used by the employees of the customer organizations and the own employees of the service provider organization. The table 5.4 lists all the different access levels. A user which has a customerrole with the role *CustomerRepresentative* can access the legalentity defined in the same customerrole definition. Also the user has to have *Customer* role in the attribute roles to identity the user as a customer user.

Organization's employees with the role *Employee* and the group membership *Sales* can see and edit data related to regular customers. Users with the role *Employee* and the group membership *SalesKeyAccount* can access and edit data related to high priority customers. The algorithm 1 in the scenario 1 can be used to check if an employee of the organization has the required role and group membership to access the resource, but it must be modified to check if the customer is a high priority customer. The check for the group membership must use correct group id depending on the result.

Algorithm 2 Algorithm to check if a user is authorized to access a certain customer identified by entityId

```
function customerUserAuthz(cRole, aRole, legalEntityId, user)
    customeruser ← user[urn : korkeala : params : scim : schemas : extension : customer : 2.0 : User]
    customerroles ← customeruser[customerroles]
    roles ← user[roles]
    for all cr in roles do                          ▷ Check if user is a customer user
        if cr[value] = cRole then
            for all c in customerroles do  ▷ Check if user has required access role
                if c[legalEntityExternalId] = legalEntityId then
                    for all r in c[roles] do
                        if r = aRole then
                            return true
                        end if
                    end for
                    return false
                end if
            end for
            return false
        end if
    end for
    return false
end function
function employeeUserAuthz(roleName, groupId, user)
    Function is the same as the function employeeUserAuthz in the algorithm 1
end function
```

Table 5.4: CRM service's access levels

Access level	Role	Customerrole	Group	Description
Customer user	Customer	Customer-Representative	-	Can see and edit customer contact information.
Organization's sales team	Employee	-	Sales	Sees all data related regular customers.
Organization's key account sales team	Employee	-	SalesKeyaccount	Sees high priority customer data.

5.2.3 Scenario 3

The third scenario does not have any new requirements for the user's attributes, but for access rights there are new requirements, the numbers 301, 302 and 303 (see the table 2.5). These requirements are about restricting permissions from a role to certain objects. These objects are explicitly attached to the access right. The SCIM "User resource schema" has attribute entitlements which have the following description: "An entitlement may be an additional right to a thing, object, or service." (Hunt et al. 2015b, p. 24). At first it seems to fulfill the requirement, but the entitlements attribute does not have sub-attributes to explicitly allow stating who is the owner of the entitlement nor does it allow linking the entitlement to a certain role. As these are required (the requirement nro. 302), entitlements can not be used for this requirement. For authorization models where an implicit association is enough, using *entitlements*-attribute is possible. By using the type and value attributes of the entitlement, the service might be able to search the owner by a separate query: from the customerroles attribute of the user (as defined in the scenario 2), the consuming service can search which roles the user has for the legalEntity by using the legalEntityId of the entitlements owner. This has the benefit that entitlements can be categorized to distinguish bank accounts, credit cards, insurances and so on.

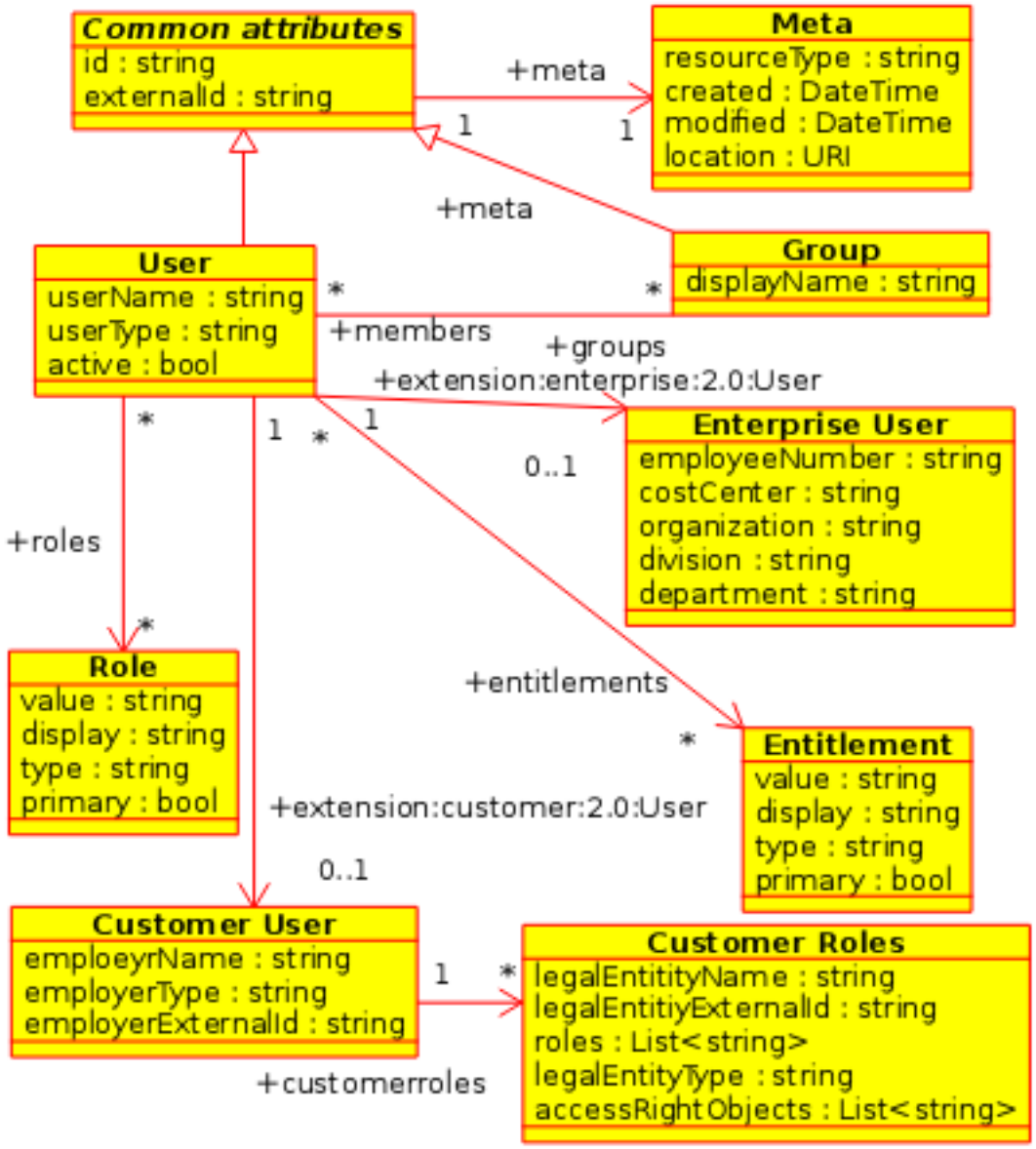

Figure 5.3: SCIM Scenario 3 UML class diagram for access rights

As the entitlements attribute does not fulfill the requirements, a new sub-attribute to the customerroles is added. This is a multi-value string attribute to express the objects the user can access with the roles that are defined in the same *customerrole* definition (satisfies the requirement nro 303). The owner of the accessRightObject(s) can be deduced from the customerrole's legalEntity attributes (satisfies the requirement nro 302). Here is the definition for the accessRightObjects-attributes, which is added to the *customerroles* attribute, defined in the previous scenario. See the figure 5.3 for a class diagram representing the scenario 3 entities.

Listing 5.2: "AccessRightObject attribute definition for customerroles"

```
{
 "name" : "accessRightObjects",
```

```
    "type" : "string",
    "multiValued" : true,
    "required" : false,
    "description" : "List of additional object user can
       access
     with this customer.",
    "mutability" : "read",
    "returned" : "default",
    "uniqueness" : "none"
  }
```

In a banking industry these objects can be bank accounts, credit cards and so on. As the customerroles is a complex attribute, its sub-attribute must not contain complex sub-attributes (Hunt et al. 2015b, p. 10), so a type or a description attribute can not be added to the accessRightObject. This might lead to collisions if the same accessRightObject value is in multiple different accessRightObject types, like in credit cards and bank accounts.

The algorithm 3 checks if a user is authorized to access an accessRightObject. As in the scenario 2 example algorithm, this takes into account that the user can be an employee of the service provider organization or a customer user. In the former case it is checked that the user has the required role and group membership (the function *customerUserAuthz* in the algorithm 3) and in the latter it checks that the user has the required role, accessRightObject and legalEntity (owner of the accessRightObject) in the same customerrole definition (the function *customerAuthz*). The accessRightObject, the role to identify customer user, the access role and the owner of the accessRightObject (entityId) in addition to the user are given as parameters for this function.

Example Service: Accounts

The example service for this scenario is called *accounts*. It is a service to manage customers' bank accounts. Customers' users with a role *AccountAdmin* in customerrole and a role *Customer* in roles can see the accounts, which are attached to the customerrole as accessRightObject. Organization's employees, which are members of the group *AccountService* and have the role *Employee* in roles, can see the accounts of regular customers. Organization's employees, which are members of the group *AccountServiceKeyAccount* can access the accounts of high priority customers. The algorithm 3 below can be used by the service to check if a user has the required access rights to access a bank account. However, it needs to be modified so that in the case of organization's employees it checks whether the owner of the bank account

Algorithm 3 Algorithm to check if a user is authorized to access an accessRightObject identified by accessRightObjectId and whose owner is identified by entityId

```
function customerUserAuthz(cRole, aRole, entityId, accessRightObj, user)
    customeruser ← user[urn : korkeala : params : scim : schemas : extension : customer : 2.0 : User]
    customerroles ← customeruser[customerroles]
    roles ← user[roles]
    for all cR in roles do                          ▷ Check if user is a customer user
        if cR = cRole then
            for all c in customerroles do ▷ Check if user has required access right
                if c[legalEntityExternalId] = entityId then
                    for all aR in c[roles] do
                        if aR = aRole then
                            for all a in c[accessRightObjects] do
                                if a = accessRightObj then
                                    return true
                                end if
                            end for
                            return false
                        end if
                    end for
                    return false
                end if
            end for
            return false
        end if
    end for
    return false
end function
function employeeUserAuthz(roleName, groupId, user)
    Function is the same as the function employeeUserAuthz in the algorithm 1
end function
```

is a high priority or a regular customer. Depending on the result it must check for the relevant group membership.

Table 5.5: Access levels for service accounts

Access level	Role	Customerrole	Group	Description
Customer user	Customer	AccountAdmin	-	Can manage customer's bank account.
Organization's account service team	Employee	-	AccountService	Sees all data related regular customers.
Organization's key account service team	Employee	-	AccountService-KeyAccount	Sees high priority customer data.

5.3 Findings

This chapter showed how the three scenarios presented in the chapter 2 can be modeled using SCIM. The presented models were able to fulfill all the user attribute, access rights and functional requirements that were set in the scenarios. This demonstrates that SCIM can be used to provide information about users and their access rights to consuming services. The presented example algorithms show how the consuming services can make authorization decisions based on the response from the SCIM service, thus demonstrating usage of SCIM for access control.

Two of the scenarios, the scenarios 2 and 3, required extending the SCIM schema in order to provide multi-tenancy access rights for users and to add employer information for the users of the customers. The SCIM standard extension mechanism proved to be useful and it can be used to extend SCIM to different kind of use cases, including serving information about external and internal users. Also it shows that organizations using SCIM should be ready to use extensions to fit their use. The restriction that complex attributes can not have a complex attribute as a sub-attribute reduces expressiveness of SCIM. This can be seen in the scenario 3 model in which the accessRightObject-attribute can not have for example a type or a displayName attribute.

6. PROOF-OF-CONCEPT

This chapter presents a proof-of-concept implementation of a SCIM service based on the models that were presented in the chapter 5. The purpose of the proof-of-concept is to assure that the models made in chapter 5 provide sufficient information to the consuming services and that they can be used to implement the scenarios. Implementation also validates the example algorithms by confirming that they can be used to decide if a user is allowed to access a resource. The test setup and details about implementation are presented first after which the implementation is evaluated.

6.1 Setup

A SCIM service and three other REST based services were included in the implementation. The three other services use the SCIM service for user identity and access right information. The services are the example services described in chapter 5. The figure 6.1 shows the deployment of the services. There is a dedicated Fedora 23[1] server which has a NGinx [2] daemon. The daemon provides HTTPS-service for all the domains and acts as a proxy server in front of the services. All the services have their own domain name, shown in the figure 6.1. All of the REST service run on Jetty-servlet engine[3] and provide a HTTP-service for the NGinx daemon. All the services have a REST interface to access the service. In addition to the REST service, there is a user interface to the service implemented using Swagger[4]. The swagger based user interface allows API documentation for the service and it also enables sending HTTP-requests to the service which helps performing exploratory tests. The figure 6.1 presents the dependencies of the services: accounts, intranet and crm services have dependency to the SCIM HTTPS service. The services were written with Clojure-programming language. Implementation used buddy[5] library for authentication and for cryptographic algorithms. See the table 6.1 for the libraries that were used in the SCIM service implementation.

[1]https://getfedora.org/
[2]https://www.nginx.com/
[3]http://www.eclipse.org/jetty/
[4]http://swagger.io/
[5]https://github.com/funcool/buddy

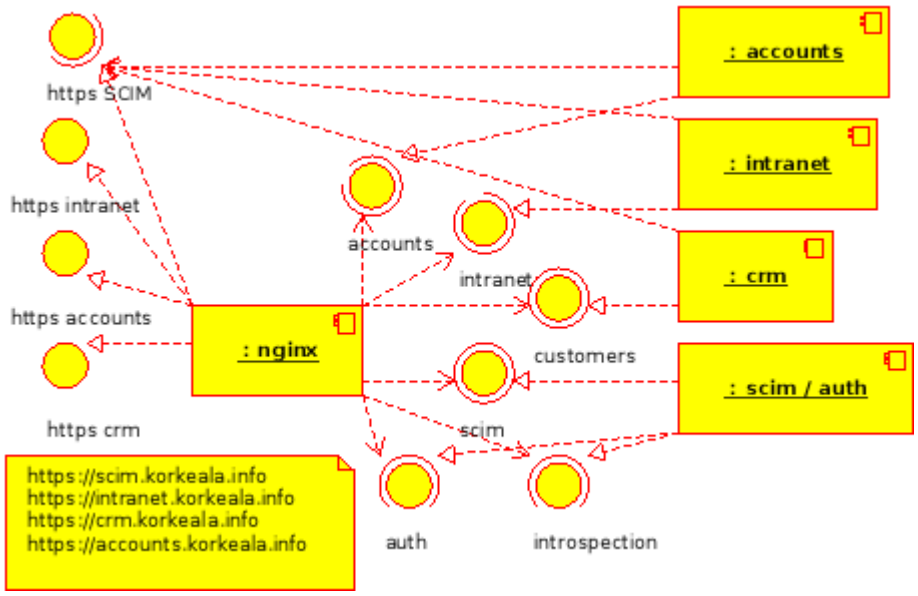

Figure 6.1: Deployment diagram of the components and their dependencies

Table 6.1: The libraries used in the SCIM service implementation

Name	Version	Homepage
clojure	1.7.0	http://www.clojure.org
compojure-api	1.0.2	https://github.com/metosin/compojure-api
compojure	1.4.0	https://github.com/weavejester/compojure
data.json	0.2.6	https://github.com/clojure/data.json
component	0.3.1	https://github.com/stuartsierra/component
ring-jetty-adapter	1.3.2	https://github.com/ring-clojure/ring
timbre	4.3.1	https://github.com/ptaoussanis/timbre
ring-defaults	0.1.5	https://github.com/ring-clojure/ring-defaults
ring-json	0.4.0	https://github.com/ring-clojure/ring-json
buddy-sign	0.12.0	https://github.com/funcool/buddy
buddy-hashers	0.14.0	https://github.com/funcool/buddy
buddy-auth	0.12.0	https://github.com/funcool/buddy
environ	1.0.2	https://github.com/weavejester/environ

6.1.1 Authentication

The test setup includes a token based authentication service. It is based on OAuth2.0 Bearer tokens and JSON Web Token (JWT) standard. The tokens are encrypted using JSON Web Signature (JWS), part of the JWT-standard. The OAuth2-service has two endpoints, one for authentication and one for token validity verification.

Authentication requires a valid username and password. On successful authentication, a token is returned in the attribute *token*. For the convenience in testing, the returned token does not in this case have an expiration date. Adding an expiration date or other claims would require passing extra parameters for the token creation function[6].

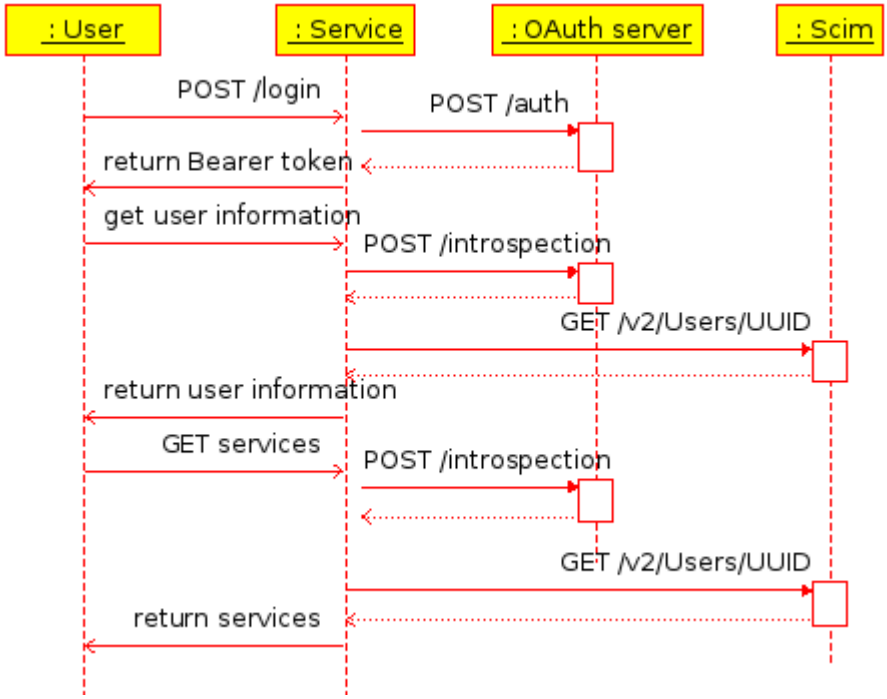

Figure 6.2: Sequence diagram when a user logs in

After authentication, the tokens can be used to access the services. There is a verification endpoint in order for the services to check if the token sent with the request is valid. It is based on the ”OAuth 2.0 Token Introspection” standard. The service expects POST request which has the token in *token* attribute (Richer 2015). On a valid token, the service responds with the following attributes; active (set to true), username and UUID. The UUID is added as a service-specific extension, it is not part of the standard OAuth2 Introspection. It helps services retrieve users from the SCIM service as the SCIM resource's id is the UUID. If an invalid token is sent to the introspection endpoint it will respond with the attribute *active* which is set to false.

[6]https://funcool.github.io/buddy-sign/latest/#claims-validation

Table 6.2: Authentication service API

Endpoint	HTTP method	Response
/oauth/auth	POST	bearer token in attribute *token*, username in attribute *username* and uuid in attribute *uuid*
/oauth/introspection	POST	username is attribute *username*, validity of the token in attribute *active* and uuid in attribute *uuid*

6.1.2 Authorization

Authorization decisions are made in the services. On incoming requests, the service first validates the OAuth service's introspection endpoint. If the token is valid, the endpoint returns user's username and UUID. The UUID is then used to retrieve user's information from the SCIM service. The service then parses the user's access rights from the response for the service and makes an authorization decision. The figure 6.3 presents a sequence diagram of the authorization process.

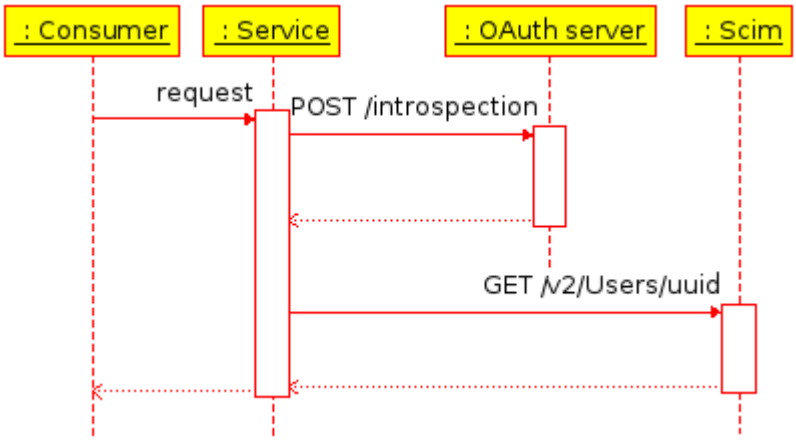

Figure 6.3: Sequence diagram on how the services authorize incoming requests

6.1.3 Testing

Testing was done by automated tests and exploratory testing. The SCIM service had unit tests and then automated integration tests were written to test all the services. The tests included positive and negative tests so that they check that the user has access to a resources she is entitled to and that they check that the user does not have access to a resource she is not entitled to.

The test data included 10 users. Access rights for those users were designed to have all the different access levels for all the services that were presented in the tables 5.3, 5.4 and 5.5. Integration tests were then written to test each of the access levels. The positive tests checked that the service responds with HTTP code 200 and returns the requested resource. The negative tests check that the service responds with 403 if the user is not entitled to the resource or with 404 if the requested resource is not found. Test suite had 120 assertions for all the services (see the table 6.3 how tests were spread for different services).

Table 6.3: Test cases and assertions

Service name	Test cases	Assertions
Scim	6	29
Intranet	6	18
CRM	8	41
Accounts	6	32
Total	26	120

6.2 Evaluation

Evaluation of the implementation is discussed next. The tests were able to validate that all the requirements were fulfilled for all the scenarios. Also, the authentication and authorization modules in the consuming services required around 100 lines of code, which indicates that the implementation is not complex. Security and and performance of the implementation are also looked at.

6.2.1 Requirements

The requirements for the scenarios were presented in chapter 2. With the presentation of the models, it was also explained how the models fulfill the requirements. The models specified in chapter 5 were implemented and the tests validated that the implementation adheres to the requirements. As all the requirements were fulfilled in the implementation, the models can be considered valid and the SCIM standard fit to be used to provide user identity service.

6.2.2 Complexity

The models for presenting users access rights were able to encode all required information and still remain quite simple for processing. The example algorithms presented in the chapter 5 and the ones used in the implementation were straightforward to implement. In the conversion from JSON representation of the user to Clojure

data structures, the lists (multi-value attributes) become vectors and JSON objects (complex attributes) become maps. Algorithms can loop through the list and get values from the map with their keys (attribute names). Also the advantage of getting all the attributes of the user with one request from the SCIM service helps processing authorization decisions. The table 6.4 lists lines of clojure code (comments, docstrings and empty lines were ignored) that were in the authentication and authorization modules of each service.

Table 6.4: Lines of clojure code in services

Service name	Authentication	Authorization
intranet	auth.clj 80 LOC	authz.clj 59 LOC
crm	auth.clj 80 LOC	authz.clj 118 LOC
accounts	auth.clj 80 LOC	authz.clj 121 LOC

However, in the scenarios 2 and 3 the access rights are defined in many different attributes: roles, groups and Customer User extension. This adds complexity and confusion on which attribute is to be used when checking user's access rights. Also, because of this the algorithms must loop through many attributes. A better model would encode all the role information in a single place.

6.2.3 Security

The SCIM schema standard has a separate chapter for security (Hunt et al. 2015b, p. 91-92) and the SCIM protocol standard has one as well (Hunt et al. 2015a, p. 77-81). It defines the security practices that should be followed, for example not storing password information in cleartext and using a secure communication channel as the information is sensitive.

The token based authentication relies solely on the token, which was used in this implementation. Everyone who has access to the token can access the services on behalf of the user. This underlines the importance of using a secure transmission method. Implementation does not set expiration for the tokens, but as noted this would be trivial to add.

No formal security testing was conducted, but the recommended security practices were followed in the implementation. These included storing passwords encrypted form and using a secure HTTPS connection. The implementation used "BCrypt password hasher combined with sha512" [7] to store passwords. The password hashing function used salt to increase security. All the services were available only over a secure HTTPS connection. This protects the user information that is transmitted,

[7] http://funcool.github.io/buddy-hashers/latest/

but also the tokens which were transferred. For a real world case, security and penetration testing is recommended as the information processed and stored is sensitive.

6.2.4 Performance

No performance tests were made. The services had to do two requests to get user's access rights information for every request they process (see the figure 6.3): first to validate token in the OAuth-service, then to retrieve user information from the SCIM service. To improve the performance in case of performance issues, caching the user information is one option. Web applications can save the user information from the SCIM service to user's session or use some external cache server, like Redis[8].

Another option to increase performance is saving the user information from the SCIM service to the token when the user logs in. Then the introspection service could return the user information from the token when consuming services validate tokens. This reduces the number of requests the consuming services have to make to process incoming requests from two to one. It also increases the size of the tokens. This option must also take into account that if user's access rights or some other information changes, the token in use must be updated.

The service implementation does not have a user session, instead it is stored in the token. This makes it easy to scale the SCIM service by installing new servers and having a load balancer in front of the application servers to route traffic to the SCIM services. The implementation had a proxy server (NGinx) in front the application servers (Jetty), which can act as a load balancer.

6.3 Assessment of SCIM Usage

For the requirements presented in the chapter 2, the SCIM standard provided a large set of attributes for users and attributes to represent access rights. These are sufficient for basic usage representing the users of an organization. The extension mechanism for SCIM allows it to adopt more specialized use cases, for example more fine-grained access rights. A downside is that the User schema's roles and entitlements attributes can not be extended. More fine-grained access rights must have new attributes or resource types, which might lead to over-lapping functionality on multiple attributes and to complexity of the access rights model.

An organization can use a similar approach that was used in this study to assess SCIM usage for their (on-premises) IAM solution to provide user information services to other services. In this thesis, the requirements were divided to three categories: access rights, attribute and functional requirements. These can be used in steps to assess requirements for the user information service (see the figure 6.4 for the steps

[8]http://redis.io

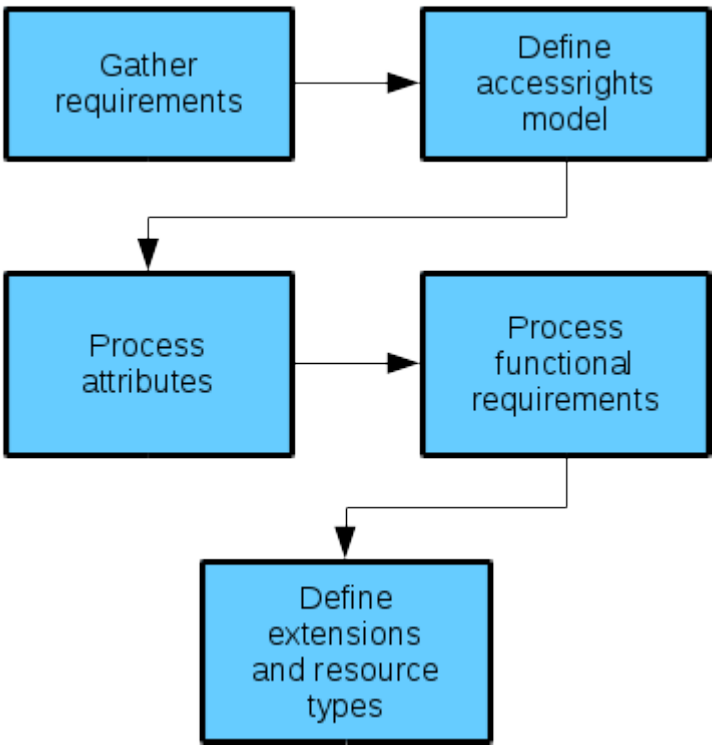

Figure 6.4: The steps for assessing whether SCIM is suitable for an organization

outline). Short instructions for each step are found in the appendix C.

7. FURTHER STUDIES

This chapter presents three interesting possibilities for future research on SCIM. One is providing general solutions to the features the protocol leaves for SCIM service providers to implement. The second is the integration of the SCIM service to an existing IAM software and the third is studying the different SCIM extensions that are being developed and the problems they try to solve.

The SCIM standard leaves some aspects of the protocol for service providers to decide and implement. One example is the authorization model for the SCIM service. One research topic would be to study common use cases for these features and implementing solutions for them.

One aspect this study did not research is integrating SCIM API for existing IAM solutions. The figure 5.1 shows one possible architectural solution, but it did not touch any of the implementation details. This could be done using *Adapter* design pattern: the SCIM API component exposes the SCIM API and uses IAM software's API to add, retrieve, modify and delete resources accordingly to the requests to the SCIM API component (Gamma et al. 1993, p. 4). Also the SCIM entity model must be mapped to the IAM software's model.

SCIM tries to solve common problems and then provide a way to extend the standard for more specific use cases. It would be interesting to see what kind of extensions are being developed, what are the major problems the extensions are trying to solve and are they overlapping with each other. This could also lead to integrating common extensions to the actual standard or referred from the standard in a particular use case.

8. CONCLUSIONS

In the beginning of this thesis two research questions were presented. These were:

- What are the advantages and disadvantages of using REST compared to SOAP for SOA based solutions?
- Does SCIM provide needed operations and entity model representation for users and their access rights?

Literary review was done to answer the first question. Based on the review, the REST based solutions have advantage in performance compared to SOAP based counterparts. Studies measured the throughout put when sending requests to service implemented in REST and SOA. Better throughout put of the REST based services was more obvious when network latency was minimized and the test client was in the same computer as the service. In test setups that used services over network, the network latency increased and the throughout put of the service part was smaller in percentage. The studies stated the performance issue was due to complexities in SOAP and its heavier use of XML. Also REST is data format independent: REST based services can use suitable data format for the task. SOAP on the other hand uses XML as its data format. In the future, REST based services can be implemented to use new, emerging data formats, which is not possible with SOAP in its current format.

The review raised concerns that frameworks and tools to implement REST services were not mature enough compared to the ones used to implement SOAP based services. Also different REST frameworks produce different kind of solutions, which might lead to compatibility problems. One explanation for these is that SOAP is an older technology and it is very standardized which helps interoperability. REST on the other hand is an emerging technology and it is more based on conventions than on standards. For these reasons, this study recommends that an organization developing REST based services should have experienced developers and it should provide implementation standards for the developer teams. However, this thesis does not consider these concerns that critical that they would prohibit the usage of REST instead of SOAP. Nonetheless, they should be taken into consideration when deciding which technology to use. This thesis provided list of questions that an organization can use in their decision making process. The questions highlight some of the important matters that should be taken into consideration.

The second research question was studied using scenarios from realworld application use cases. Three scenarios were presented and then they were modeled using SCIM. They had requirements on what kind of attributes user must have and what kind of access rights model is required. Also an example service was presented for each of the scenarios. The first scenario was a service for an organization's internal use, which requires role-based and group-based access rights information to make authorization decisions. The second scenario requires access right model to have a new dimension: customer information. In the scenario an organization provides services to its customers and customer's users can have different access rights to different customers. The access rights model must provide information to which customer the access right is valid. The third scenario needed more fine-grained access rights: role-based model can have restrictions on which objects users are allowed to access with a given role.

The scenarios were modeled with SCIM. The first scenario did not require extensions to SCIM: SCIM core resource schema was expressive enough to implement all the requirements of the scenario. The second and the third scenario required extending the SCIM schema to implement all requirements for the scenarios. This was accomplished using SCIM schema extension capabilities. The extensions were needed to express user's access rights for different organizations in the scenario 2 and to express restrictions to roles in the scenario 3. Also, in order to state the employer of the user, new attributes were needed as Enterprise User-extension did not have an attribute that would distinctively express the employer organization. The extensions added complexity to the two models as access rights were defined in multiple attributes. This makes the models harder to understand and adds confusion about which access right attribute is used for which purpose. Each of the scenario models also included an example algorithm to check if a user has the required access rights to a resource.

Models and algorithms were validated in a proof-of-concept implementation. The proof-of-concept included a SCIM service and one REST based service for each of the scenarios. The three implemented services used the SCIM service to query user identity and access rights information. In each case, the service was able to make authorization decisions based on the data the SCIM service provided and the users were able to access resources to which they had access rights, but not to the ones they did not have access to. This indicates that the presented models and algorithms were valid.

An organization can use a similar approach that was used in this thesis to assess whether SCIM is suitable for their use. Short instructions for the five step approach were provided in the appendix C.

This thesis concludes that the SCIM standard can be used to provide user identity and access right information to other services. By using the existing standards for the services they build and/or provide, organizations can get benefits and reduce costs. As the SCIM standard is designed to help user identity provisioning to cloud services, organizations can also open the SCIM service to their customers and let them connect their IAM software to the SCIM service. This allows the customer to automatically provision their employee information and so help them manage their users' identity and access rights information. This might give competitive advantage to the organization on the market.

BIBLIOGRAPHY

Aihkisalo, T. and Paaso, T. 2012. Latencies of service invocation and processing of the rest and soap web service interfaces. In Services (SERVICES), 2012 IEEE Eighth World Congress on, pages 100–107.

Arsanjani, A. 2004. Service-oriented modeling and architecture. IBM developer works, pages 1–15.

Bohren, J., Boucher, R., Cohen, D., Cole, G., Collingham, C., Elron, R., Fanti, M., Glazer, I., Hu, J., Jacobsen, R., Larson, J., Lockhart, H., Mishra, P., Raepple, M., Rolls, D., Spaulding, K., Sodhi, G., Williams, C., and Woods, G. 2006. Oasis service provisioning markup language (spml) version 2. [WWW]. Retreived 15.4.2016. Available at: http://www.oasis-open.org/committees/download.php/17708/pstc-spml-2.0-os.zip.

Celesti, A., Tusa, F., Villari, M., and Puliafito, A. 2010. Security and cloud computing: Intercloud identity management infrastructure. In Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on, pages 263–265.

Drake, T., Mortimore, C., Ansari, M., Grizzle, K., and Wahlström, E. 2012. System for cross-domain identity management:protocol 1.1. [WWW]. Retreived 21.10.2015. Available at: http://www.simplecloud.info/specs/draft-scim-api-01.html.

Emig, C., Brandt, F., Kreuzer, S., and Abeck, S. 2007. Identity as a Service – Towards a Service-Oriented Identity Management Architecture, volume 4606 of Lecture Notes in Computer Science, pages 1–8. Springer Berlin Heidelberg. 10.1007/978-3-540-73530-4_1.

Feuerlicht, G. 2010. Advances in Intelligent Web Mastering - 2: Proceedings of the 6th Atlantic Web Intelligence Conference - AWIC'2009, Prague, Czech Republic, September, 2009, chapter Next Generation SOA: Can SOA Survive Cloud Computing?, pages 19–29. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fielding, R. T. 2000. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1993. Design patterns: Abstraction and reuse of object-oriented design. In European Conference on Object-Oriented Programming, pages 406–431. Springer.

Gorski, P. L., Lo Iacono, L., Nguyen, H. V., and Torkian, D. B. 2014. Service-Oriented and Cloud Computing: Third European Conference, ESOCC 2014, Manchester, UK, September 2-4, 2014. Proceedings, chapter SOA-Readiness of REST, pages 81–92. Springer Berlin Heidelberg, Berlin, Heidelberg.

Groba, C., Braun, I., Springer, T., and Wollschlaeger, M. 2008. A service-oriented approach for increasing flexibility in manufacturing. In Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on, pages 415–422. IEEE.

Gudgin, M., Hadley, M., Mendelsohn, N., Lafon, Y., Moreau, J., Karmarkar, A., and Nielsen, H. 2007. Soap version 1.2 part 1: Messaging framework, 2nd edn. [WWW]. Retrieved 14.7.2016. Available at: https://www.w3.org/TR/soap12/.

Hunt, P., Grizzle, K., Ansari, M., Wahlström, E., and Mortimore, C. 2015a. System for Cross-domain Identity Management: Protocol. RFC 7644, Internet Engineering Task Force.

Hunt, P., Grizzle, K., Wahlström, E., and Mortimore, C. 2015b. System for Cross-domain Identity Management: Core Schema. RFC 7643, Internet Engineering Task Force.

Kumari, S. and Rath, S. K. 2015. Performance comparison of soap and rest based web services for enterprise application integration. In Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on, pages 1656–1660.

Linden, M. 2009. Organisational and cross-organisational identity management.

Mortimore, C., Harding, P., Madsen, P., and Drake, T. 2012. System for cross-domain identity management: Core schema 1.1. Technical Report 7643. [WWW]. Retreived 21.10.2015. Available at: http://www.simplecloud.info/specs/draft-scim-core-schema-01.html.

Pautasso, C., Zimmermann, O., and Leymann, F. 2008. Restful web services vs. big'web services: making the right architectural decision. In Proceedings of the 17th international conference on World Wide Web, pages 805–814. ACM.

Peng, D., Li, C., and Huo, H. 2009. An extended usernametoken-based approach for rest-style web service security authentication. In Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, pages 582–586. IEEE.

Reed, A., Cohen, D., Sodhi, G., Woods, G., Lockhart, H., Bohren, J., Larson, J., Fernandez, J., Leibmann, M., Polan, M., Madsen, P., Elron, R., Gallotta, T., and Kirsh, Y. 2003. Service provisioning markup language (spml) version 1.0. [WWW]. Retreived 15.4.2016. Available at: https://www.oasis-open.org/committees/download.php/4137/os-pstc-spml-core-1.0.pdf.

Richer, J. 2015. Oauth 2.0 token introspection. RFC 7662. [WWW]. Retreived 9.5.2016. Available at: https://tools.ietf.org/html/rfc7662.

Serme, G., Oliveira, A. D., Massiera, J., and Roudier, Y. 2012. Enabling message security for restful services. In Web Services (ICWS), 2012 IEEE 19th International Conference on, pages 114–121.

Spencer, T. 2012. Identity in the cloud. Computer Fraud & Security, 2012(7):19 – 20.

Sward, R. E. and Whitacre, K. J. 2008. A multi-language service-oriented architecture using an enterprise service bus. ACM SIGAda Ada Letters, 28(3):85–90.

A. SCIM SCHEMA EXTENSIONS

Definitions for the SCIM extensions for the User schema. This extension adds customers' users link to their employer and list of permissions for different customers.

```
{
 "id" : "urn:korkeala:params:scim:schemas:extension:
    customer:2.0:User",
 "name" : "Customer User",
 "attributes" : [
       {
       "name" : "employerExternalId",
       "type" : "string",
       "multiValued" : false,
       "required" : false,
       "mutability" : "read",
       "returned" : "default",
       "uniqueness" : "none",
       "description" : "Numeric or alphanumeric identifier
       assigned to a user's employer, for example business
       id or social security number."
       },
       {
       "name" : "employerName",
       "type" : "string",
       "multiValued" : false,
       "required" : false,
       "description" : "Name of user's employer.",
       "mutability" : "read",
       "returned" : "default",
       "uniqueness" : "none"
       },
       {
       "name" : "employerType",
       "type" : "string",
```

```
      "multiValued" : false,
      "required" : false,
      "description" : "Type of the user's employer,
      person or organization.",
      "mutability" : "read",
      "returned" : "default",
      "uniqueness" : "none"
      },
  {
  "name" : "customerroles",
  "type" : "complex",
  "multiValued" : true,
  "description" : "Accessrights for legal entities
  for multi-tenancy setup.",
  "required" : false,
  "mutability" : "read",
  "returned" : "default",
  "uniqueness" : "none",
  "subAttributes" : [
  {
    "name" : "legalEntityExternalId",
    "type" : "string",
    "multiValued" : false,
    "required" : false,
    "mutability" : "read",
    "returned" : "default",
    "uniqueness" : "none",
    "description" : "Numeric or alphanumeric identifier
    assigned to a legal entity, for example business id
    or social security number."
  },
  {
    "name" : "legalEntityName",
    "type" : "string",
    "multiValued" : false,
    "required" : false,
    "description" : "Name of the legal entity.",
    "mutability" : "read",
    "returned" : "default",
```

```
      "uniqueness" : "none"
    },
    {
      "name" : "legalEntityType",
      "type" : "string",
      "multiValued" : false,
      "required" : false,
      "description" : "Type of the legal entity,
      person or organization.",
      "mutability" : "read",
      "returned" : "default",
      "uniqueness" : "none"
    },
    {
      "name" : "roles",
      "type" : "string",
      "multiValued" : true,
      "required" : false,
      "description" : "List of role user can access
      representing the legal entity.",
      "mutability" : "read",
      "returned" : "default",
      "uniqueness" : "none",
    }
    ]
    }
 ]
}
```

Listing A.1: ”Customer user extension to the User schema for the scenario 2”

The SCIM extension for representing permissions to customers for the scenario 3.

```
{
 "id" : "urn:korkeala:params:scim:schemas:extension:
    customer:2.0:User",
 "name" : "Customer User",
 "attributes" : [
    {
    "name" : "employerExternalId",
```

```
"type" : "string",
"multiValued" : false,
"required" : false,
"mutability" : "read",
"returned" : "default",
"uniqueness" : "none",
"description" : "Numeric or alphanumeric identifier
assigned to a user's employer, for example business
id or social security number."
},
{
"name" : "employerName",
"type" : "string",
"multiValued" : false,
"required" : false,
"description" : "Name of user's employer.",
"mutability" : "read",
"returned" : "default",
"uniqueness" : "none"
},
{
"name" : "employerType",
"type" : "string",
"multiValued" : false,
"required" : false,
"description" : "Type of the user's employer,
person or organization.",
"mutability" : "read",
"returned" : "default",
"uniqueness" : "none"
},

{
"name" : "customerroles",
"type" : "complex",
"multiValued" : true,
"description" : "Accessrights for legal entities
for multi-tenancy setup.",
"required" : false,
```

```
"mutability" : "read",
"returned" : "default",
"uniqueness" : "none",
"subAttributes" : [
{
  "name" : "legalEntityExternalId",
  "type" : "string",
  "multiValued" : false,
  "required" : false,
  "mutability" : "read",
  "returned" : "default",
  "uniqueness" : "none",
  "description" : "Numeric or alphanumeric identifier
     assigned
  to a legal entity, might be business id or social
     security number."
},
{
  "name" : "legalEntityName",
  "type" : "string",
  "multiValued" : false,
  "required" : false,
  "description" : "Name of the legal entity.",
  "mutability" : "read",
  "returned" : "default",
  "uniqueness" : "none"
},
{
  "name" : "legalEntityType",
  "type" : "string",
  "multiValued" : false,
  "required" : false,
  "description" : "Type of the legal entity, person or
     organization.",
  "mutability" : "read",
  "returned" : "default",
  "uniqueness" : "none"
},
{
```

```
      "name" : "roles",
      "type" : "string",
      "multiValued" : true,
      "required" : false,
      "description" : "List of role user can access
      representing the legal entity.",
      "mutability" : "read",
      "returned" : "default",
      "uniqueness" : "none",
    },
    {
      "name" : "accessRightObjects",
      "type" : "string",
      "multiValued" : true,
      "required" : false,
      "description" : "List of additional object user
      can access for this legal entity.",
      "mutability" : "read",
      "returned" : "default",
      "uniqueness" : "none"
    }
    ]
    }
 ]
}
```

Listing A.2: ”Customer user extension to the User schema for the scenario 3”

B. SCIM USER EXAMPLES

Here are example responses from the implemented SCIM service. First one was used to test the scenario 1, it represents an employee of the organization.

```
$ http -v  --json GET https://scim.korkeala.info/scim/v2/
   Users/24cd6248-7da8-44df-befe-7e42e1799d88 'Content-Type
   :application/json' 'Authorization:Bearer <token>'
GET /scim/v2/Users/24cd6248-7da8-44df-befe-7e42e1799d88
   HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Authorization: Bearer <token>
Connection: keep-alive
Content-Type: application/json
Host: scim.korkeala.info
User-Agent: HTTPie/0.9.2


HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 1627
Content-Type: application/json; charset=utf-8
Date: Tue, 21 Jun 2016 13:31:59 GMT
Server: nginx/1.8.1
Strict-Transport-Security: max-age=31536000;
   includeSubdomains

{
    "active": true,
    "addresses": [
        {
            "country": "FI",
            "locality": "Helsinki",
            "postalCode": "00100",
```

```
            "streetAddress": "Mannerheimintie 1"
        }
    ],
    "emails": [
        {
            "primary": true,
            "type": "work",
            "value": "ppetterson@invalid.com"
        }
    ],
    "externalId": "ppetterson",
    "groups": [
        {
            "$ref": "https://scim.korkeala.info/scim/v2/
               Groups/ba4fcf5b-3327-4e2e-afd3-c1f5ea41c12a"
               ,
            "display": "AccountService",
            "value": "ba4fcf5b-3327-4e2e-afd3-c1f5ea41c12a"
        },
        {
            "$ref": "https://scim.korkeala.info/scim/v2/
               Groups/5b7ea8d9-87a3-4865-b467-dd8336bdeb0d"
               ,
            "display": "AccountServiceKeyAccount",
            "value": "5b7ea8d9-87a3-4865-b467-dd8336bdeb0d"
        },
        {
            "$ref": "https://scim.korkeala.info/scim/v2/
               Groups/fae5e0e4-2f24-4c19-913c-f1f7932d15e6"
               ,
            "display": "SalesKeyAccount",
            "value": "fae5e0e4-2f24-4c19-913c-f1f7932d15e6"
        },
        {
            "$ref": "https://scim.korkeala.info/scim/v2/
               Groups/87156a4e-906f-44da-90f4-f917bebb22e5"
               ,
            "display": "Sales",
            "value": "87156a4e-906f-44da-90f4-f917bebb22e5"
```

```
        }
    ],
    "id": "24cd6248-7da8-44df-befe-7e42e1799d88",
    "meta": {
        "created": "2016-05-16T09:12:51Z",
        "lastModified": "2016-05-16T09:32:13Z",
        "location": "https://scim.korkeala.info/scim/v2/
           Users/24cd6248-7da8-44df-befe-7e42e1799d88",
        "resourceType": "User"
    },
    "name": {
        "familyName": "Petterson",
        "formatted": "Mr Petterson",
        "givenName": "Peter"
    },
    "phoneNumbers": [
        {
            "primary": true,
            "type": "work",
            "value": "tel. 555-555-6666"
        }
    ],
    "roles": [
        {
            "display": "Basic role for employees",
            "value": "Employee"
        }
    ],
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise
           :2.0:User"
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:
       User": {
        "department": "Sales",
        "employeeNumber": "324971"
    },
    "userName": "ppetterson",
```

```
    "userType": "User"
}
```

Listing B.1: ”Example user representation for the scenario 1”

The following user response from the SCIM service represents the scenario 2 user; it has access rights to two different customers.

```
$ http -v  --json GET https://scim.korkeala.info/scim/v2/
   Users/2bca798a-74da-4537-bb2c-a1da147354fc 'Content-Type
   :application/json' 'Authorization:Bearer <token>'
GET /scim/v2/Users/2bca798a-74da-4537-bb2c-a1da147354fc
   HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Authorization: Bearer <token>
Connection: keep-alive
Content-Type: application/json
Host: scim.korkeala.info
User-Agent: HTTPie/0.9.2


HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 1466
Content-Type: application/json; charset=utf-8
Date: Tue, 21 Jun 2016 13:37:56 GMT
Server: nginx/1.8.1
Strict-Transport-Security: max-age=31536000;
   includeSubdomains

{
    "active": true,
    "addresses": [
        {
            "country": "FI",
            "locality": "Helsinki",
            "postalCode": "00500",
            "streetAddress": "Hämeentie 1"
        }
    ],
```

```
    "emails": [
        {
            "primary": true,
            "type": "work",
            "value": "ddavidson@example.com"
        }
    ],
    "externalId": "ddavidson",
    "id": "2bca798a-74da-4537-bb2c-a1da147354fc",
    "meta": {
        "created": "2016-05-16T10:42:30Z",
        "lastModified": "2016-05-16T10:42:30Z",
        "location": "https://scim.korkeala.info/scim/v2/
           Users/2bca798a-74da-4537-bb2c-a1da147354fc",
        "resourceType": "User"
    },
    "name": {
        "familyName": "Davidson",
        "formatted": "Mr Davidson",
        "givenName": "David"
    },
    "phoneNumbers": [
        {
            "primary": true,
            "type": "work",
            "value": "tel. 555-666-55555"
        }
    ],
    "roles": [
        {
            "display": "Basic role for customers",
            "value": "Customer"
        }
    ],
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise
           :2.0:User",
```

```
        "urn:korkeala:params:scim:schemas:extension:
           customer:2.0:User"
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:
       User": {},
    "urn:korkeala:params:scim:schemas:extension:customer
       :2.0:User": {
        "customerroles": [
            {
                "legalEntityExternalId": "456365-4",
                "legalEntityName": "Customer Company AB Oy"
                   ,
                "legalEntityType": "Organization",
                "roles": [
                    "AccountAdmin",
                    "CustomerRepresentative"
                ]
            },
            {
                "legalEntityExternalId": "1234732-1",
                "legalEntityName": "Customer Company CD Oy"
                   ,
                "legalEntityType": "Organization",
                "roles": [
                    "CustomerRepresentative"
                ]
            }
        ],
        "employerExternalId": "29876234-7",
        "employerName": "Customer Employer GH Oy",
        "employerType": "Organization"
    },
    "userName": "ddavidson",
    "userType": "User"
}
```

Listing B.2: ”Example user representation for the scenario 2”

The following user response from the SCIM service represents the scenario 3 user; it has access rights to two accessRightsObjects to one customer and one accessRight-

sObject to another customer.

```
$ http -v  --json GET https://scim.korkeala.info/scim/v2/
   Users/1c5f78e6-4bc1-4111-8089-4ecdfeea23d9 'Content-Type
   :application/json' 'Authorization:Bearer <token>'
GET /scim/v2/Users/1c5f78e6-4bc1-4111-8089-4ecdfeea23d9
   HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Authorization: Bearer <token>
Connection: keep-alive
Content-Type: application/json
Host: scim.korkeala.info
User-Agent: HTTPie/0.9.2


HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 1577
Content-Type: application/json; charset=utf-8
Date: Tue, 21 Jun 2016 13:40:13 GMT
Server: nginx/1.8.1
Strict-Transport-Security: max-age=31536000;
   includeSubdomains

{
    "active": true,
    "addresses": [
        {
            "country": "FI",
            "locality": "Helsinki",
            "postalCode": "00550",
            "streetAddress": "Mäkelänkatu 1"
        }
    ],
    "emails": [
        {
            "primary": true,
            "type": "work",
            "value": "jjones@example.com"
```

```
        }
    ],
    "externalId": "jjones",
    "id": "1c5f78e6-4bc1-4111-8089-4ecdfeea23d9",
    "meta": {
        "created": "2016-05-16T10:42:30Z",
        "lastModified": "2016-05-16T10:42:30Z",
        "location": "https://scim.korkeala.info/scim/v2/
           Users/1c5f78e6-4bc1-4111-8089-4ecdfeea23d9",
        "resourceType": "User"
    },
    "name": {
        "familyName": "Jones",
        "formatted": "Ms Jones",
        "givenName": "Johanna"
    },
    "phoneNumbers": [
        {
            "primary": true,
            "type": "work",
            "value": "tel. 555-555-5566"
        }
    ],
    "roles": [
        {
            "display": "Basic role for customers",
            "value": "Customer"
        }
    ],
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise
           :2.0:User",
        "urn:korkeala:params:scim:schemas:extension:
           customer:2.0:User"
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:
       User": {},
```

```
    "urn:korkeala:params:scim:schemas:extension:customer
       :2.0:User": {
        "customerroles": [
            {
                "accessRightObjects": [
                    "DE 0750 8900 0000 0175 7814",
                    "DE72 3702 0500 0009 7097 00"
                ],
                "legalEntityExternalId": "456365-4",
                "legalEntityName": "Customer Company AB Oy"
                   ,
                "legalEntityType": "Organization",
                "roles": [
                    "AccountAdmin",
                    "CustomerRepresentative"
                ]
            },
            {
                "accessRightObjects": [
                    "DE44 5001 0517 5407 3249 31"
                ],
                "legalEntityExternalId": "1234732-1",
                "legalEntityName": "Customer Company CD Oy"
                   ,
                "legalEntityType": "Organization",
                "roles": [
                    "AccountAdmin"
                ]
            }
        ],
        "employerExternalId": "2134456-4",
        "employerName": "Customer Employer AB",
        "employerType": "Organization"
    },
    "userName": "jjones",
    "userType": "User"
}
```

Listing B.3: ”Example user representation for the scenario 3”

C. INSTRUCTIONS TO ASSESS SCIM USAGE

Below are short instructions for assessing possible SCIM usage for implementing user information service (see the figure 6.4 for the outline of the steps for the assessment). If during the assessment some critical requirements are not possible to fulfill with SCIM and its possible extensions, it is recommended to use some other protocol.

C.0.1 Gather Requirements

First gather requirements from the services which will use the SCIM service and list information that the consuming services need from the user information service. Also list requirements that the organization's architecture imposes and organization's services that the SCIM must integrate to. Divide requirements for access rights, for resource types attributes and functional requirements. List possible other resource types than user or group if they are required.

C.0.2 Define Model for Access rights

After gathering the requirements, define a model for the access rights. Try to implement the model using the SCIM attributes for access rights: roles, entitlements and groups. If they can not express required access rights for consuming services, try to model them using extensions, but try not to add complexity to the model by having many access rights attributes with similar functions. The models in this thesis can also be used as base for the access rights model. If there are multi-tenancy requirements, see if they fall to a category listed in the SCIM standard (Hunt et al. 2015a, p. 75-77). Also see the scenario 2 multi-tenancy requirements modelled in this thesis as an example for alternative approach.

C.0.3 Attributes for Resource Types

After having gathered the requirements, check that all the required attributes or similar ones are defined in the SCIM resource type. If some attributes are missing, they can be added through SCIM extensions.

C.0.4 Functional Requirements and Extensions

Next, check all the functional requirements. If the organization does not have REST services, use the checklist provided in the end of the chapter 3 to assess if REST technology is suitable for the organization. If the organization's current IAM solution supports SCIM, check that extensions can be defined if they are required and that new resource types can be defined if they are required. If the current solution does not support SCIM, check if the current solution has necessary APIs to develop a SCIM adapter for the IAM solution.

C.0.5 Extensions and Resource Types

After the requirements are all processed, define extensions or new resource types if there are requirements that need them. Note that multiple extensions can be defined, in other words, different types of information can have their own extensions. Also avoid redefining attributes already in the SCIM schema.