HAAGA-HELIA
ammattikorkeakoulu

Finnvalli Web Services

Pieter Starmans

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

6.2.2014

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Pieter Starmans	**Ryhmä tai aloitusvuosi** 2010
Opinnäytetyön nimi Finnvalli Web Services	**Sivu- ja liitesivumäärä** 26 + 8
Ohjaaja tai ohjaajat Markku Kuitunen	

Tämän opinnäytetyön tarkoituksena on toteuttaa Oy Finnvalli Ab:lle julkinen Web Service- rajapinta, jonka kautta voidaan siirtää XML-muotoisia aineistoja toimeksiantajan kehittämän Fivaldi sovelluksen ja toisten sovellusten välillä käyttäen SOAP- protokollaa. Opinnäytetyö käsittelee Web Service rajapinnan kehittämistä. Työssä kuvataan SOAP sanomien sekä WSDL tiedoston rakenteita ja niiden käsittelyä Web Service rajapinnassa. Työssä kerrotaan myös autentikoinnin ja authorisoinnin toteuttamisesta Web Servicessä OASIS organisaation määrittelemällä WS-Security standardin UsernameTokenilla.

Opinnäytetyössä kuvaillaan näiden edellä mainittujen asioiden toteutusta opinnäytetyön tilaajalle toteutettuun Web Service rajapintaan.

Tämä opinnäytetyö toteutettiin marraskuun 2013 – tammikuun 2014 välisenä aikana. Web Servicen kehityksessä käytettiin kehitystyökaluna Oraclen JDeveloper-työkalua ja Web Servicen toteutukseen käytettiin ohjelmointikielenä Java-ohjelmointikieltä. Rajapinnasta saadun ja sinne lähetettävän aineiston käsittely sekä autentikointi ja auktorisointi toteutettiin Oraclen tietokannassa käyttäen ohjelmointikielenä Oraclen PL/SQL- ohjelmointikieltä.

Projekti onnistui kokonaisuudessaan hyvin ja opinnäytetyön tilaajan haluama Web Service rajapinta saatiin toteutettua tilaajan vaatimusten mukaisesti.

Asiasanat
Web Services, SOAP, WSDL, WS-Security

HAAGA-HELIA
University of Applied Sciences

Abstract

6.2.2014

Programme in Information Technology

Author(s) Pieter Starmans	**Group or year of entry** 2013
The title of thesis Finnvalli Web Services	**Number of report pages and attachment pages** 26 + 8
Advisor(s) Markku Kuitunen	

The purpose of this thesis was to develop a public Web Service for Oy Finnvalli Ab. The purpose of developed Web Service is to handle SOAP- style messages transported using http- protocol between client software and developed Web Services. Inside these requests main purpose is to send and receive SOAP messages with attachments containing different XML files.

This thesis describes development of Web Services, structure and meaning of WSDL files and SOAP messages. The thesis describes logic of handling of SOAP messages in Web Services. This thesis also describes WS-Security standard UsernameToken Profile created by OASIS organization and usage of the profile in developed Web Services.

This project started at November 2013 and finished at January 2014. As a developing tool in this Web Services development was Oracle's JDeveloper. Programming language used to develop Web Services was Java. Database package for endpoint development for Web Services in current software's Oracle database was developed using PL/SQL programming language. Authentication and authorization for the Web Services was developed at endpoint in current software's database.

This project was successful and Finnvalli was satisfied with results of the project.

Key words
Web Services, SOAP, WSDL, WS-Security

Sanasto

Autentikaatio

Autentikaatio on vapaasti käännetty englannin kielen sanasta authentication. Autentikaatio tarkoittaa sitä, että käyttäjä tunnistautuu todistamalla olevansa se, joka esittää olevansa. Tässä asiayhteydessä autentikaatiolla tarkoitetaan tunnistautumista käyttäjätunnuksella ja salasanalla. (Autentikaatio.)

Auktorisointi

Auktorisointi on vapaasti käännetty englannin kielen sanasta authorization. Auktorisointi tarkoittaa sitä, että käyttäjä saa tehdä hänelle määriteltyjä asioita. Tässä asiayhteydessä authorisoinnilla tarkoitetaan autentikoidun käyttäjän käyttöoikeuksia eri yritysten eri aineistojen siirtoon. (Auktorisointi.)

HTTP (Hypertext Transfer Protocol)

HTTP on Request/Response- tiedonsiirtoprotokolla. Käyttäjä lähettää Request sanomat serverille, joka vastaa Responce sanomalla. (HTTP.)

Request

Request sanoma on käyttäjältä serverille lähettämä sanoma, jossa ilmaistaan vähintään metodi, esim. GET tai POST, osoite johon Request sanoma osoitetaan, protokollaversio sekä muita valinnaisia tietoja (Request-sanoma.)

Responce

Responce on vastaus sanoma käyttäjän lähettämään Request sanomaan. Sanomassa on sisältönä vähintään protokollaversio ja 3 numeroinen status- koodi, sekä lyhyt tekstikuvaus status koodista. Status koodin ensimmäinen numero määrittelee Responce sanoman tyylin, esim. 2xx:Success, 4xx:Client Error. (Responce-sanoma.)

Server

Ohjelma, johon voi muodostaa yhteyksiä ja käsittelee Request sanomia ja vastaa niihin Responce sanomilla (Server.)

String

String eli vapaasti suomennettuna merkkijono, on tietotyyppi, joka sisältää sanansa mukaisesti merkkejä(Java merkkijono.)

Käyttäjä

Tässä opinnäytetyössä käyttäjällä viitataan ohjelmaan, joka muodostaa yhteyden Web Serviceen lähettääkseen Request sanoman.
(Käyttäjä.)

XML (Extensible Markup Language)

XML on standardisoitu tekstimuotoinen kieli, jonka avulla voidaan kuvailla hierarkkista tietoa. XML on laajasti käytetty tiedon jakamisen kieli sovellusten välisessä tiedonsiirrossa.(XML.)

Web Service

Web Service on ohjelmisto, joka on suunniteltu ohjelmistojen väliseen kanssakäymiseen Internetin välityksellä. Sen rajapinnan kuvaus on toisten sovellusten ymmärtämässä muodossa WSDL tiedostona, jonka perusteella toiset sovellukset voivat kutsua Web Serviceä. (Web Service kuvaus.)

SOAP (Simple Object Access Protocol)

SOAP on yksinkertainen ja kevyt XML- muotoinen sanoma, joka voidaan lähettää sovellusten välillä verkon yli ohjelmointikielestä tai tiedonsiirtoprotokollasta riippumatta (SOAP protokolla.)

OASIS

OASIS on voittoa tavoittelematon organisaatio jonka määrittelemiä standardeja käytetään ja arvostetaan yleisesti esim. XML:ssä (OASIS organisaatio.)

Sisällys

1 Johdanto

1.1 Opinnäytetyön tavoitteet

Opinnäytetyöprojektin tavoitteena oli toteuttaa toimiva Web Services- rajapinta, jonka välityksellä voidaan siirtää aineistoja olemassa olevan sovelluksen ja käyttäjän välillä. Aineistojen sisältö siirretään toteutetun Web Servicen ja sitä kutsuvan sovelluksen välillä SOAP sanoman liitteenä. Tällaisen liitteellisen SOAP sanoman virallinen nimi on Soap With Attachments.

Itse aineistojen käsittely tapahtuu olemassa olevilla tietokantaproseduureilla. Tässä projektissa toteutettavan Web Servicen tehtävä on toimia rajapintana julkisen verkon ja suljetun sovelluksen ja sen tietojen välillä. Web Servicen tehtävänä on välittää autentikaatio- ja auktorisointitiedot projektissa toteutettuun tietokantapakettiin, jossa toteutetaan näiden tietojen perusteella autentikointi ja auktorisointi, sekä toimia tietojen välittäjänä autentikoidun ja auktorisoidun käyttäjän sekä tietokannan välillä.

Aineiston pyytäjän tai lähettäjän autentikointi- ja auktorisointitietojen välittäminen SOAP sanomissa toteutettiin OASIS- standardin mukaisella WS-Security Usernametokenilla. Autentikointi ja auktorisointitietojen tarkistuksen logiikka toteutettiin tietokannassa, johon tehtiin uusi PL/SQL- tietokantapaketti. Pakettiin tehtiin tarvittavat tietokantaproseduurit sekä funktiot tätä varten.

Autentikoidun sekä auktorisoidun käyttäjän lähettämän SOAP sanoman perusteella kutsutaan sovelluksen tietokannassa olevia XML-aineiston sisäänlukua ja muodostamista varten olevia proseduureja ja välitetään pyynnön tyypistä riippuen joko käyttäjän SOAP sanoman liitteenä oleva aineisto tietokantaan, tai käyttäjän pyytämät aineistot SOAP sanoman liitteenä käyttäjälle.

Suurimmat oppimistavoitteet projektissa olivat SOAP- standardien oppiminen ja sanomien käsittely, etenkin WS-Security standardien omaksuminen ja käyttö, sekä itse Web Servicen toiminnallisuuden suunnittelu ja toteutus.

1.2 Opinnäytetyön tausta

Opinnäytetyön tilaajan, Oy Finnvalli Finland Ab:n suunnitelmissa on ollut jo pitkään toteuttaa standardien mukainen Web Services- rajapinta, jonka kautta autentikoidut ja auktorisoidut käyttäjät voisivat turvallisesti vastaanottaa ja lähettää XML- muotoisia sanomia. Näitä tarvitaan mm. silloin kun asiakkaalta tulee ulkopuolisesta ohjelmistosta aineistoja yrityksen kehittämään Fivaldi-sovellukseen. Fivaldi tukee tällä hetkellä XML-aineistojen sisäänlukua ja lähetystä, mutta tiedostojen siirto halutaan tulevaisuudessa tämän projektin myötä toteuttaa Web Service- kanavalla HTTP- protokollalla SOAP-sanomina. Projekti on aiemmin toimeksi annettu opinnäytetyöksi, mutta opinnäytetyön tekijä ei onnistunut toteuttamaan projektia.

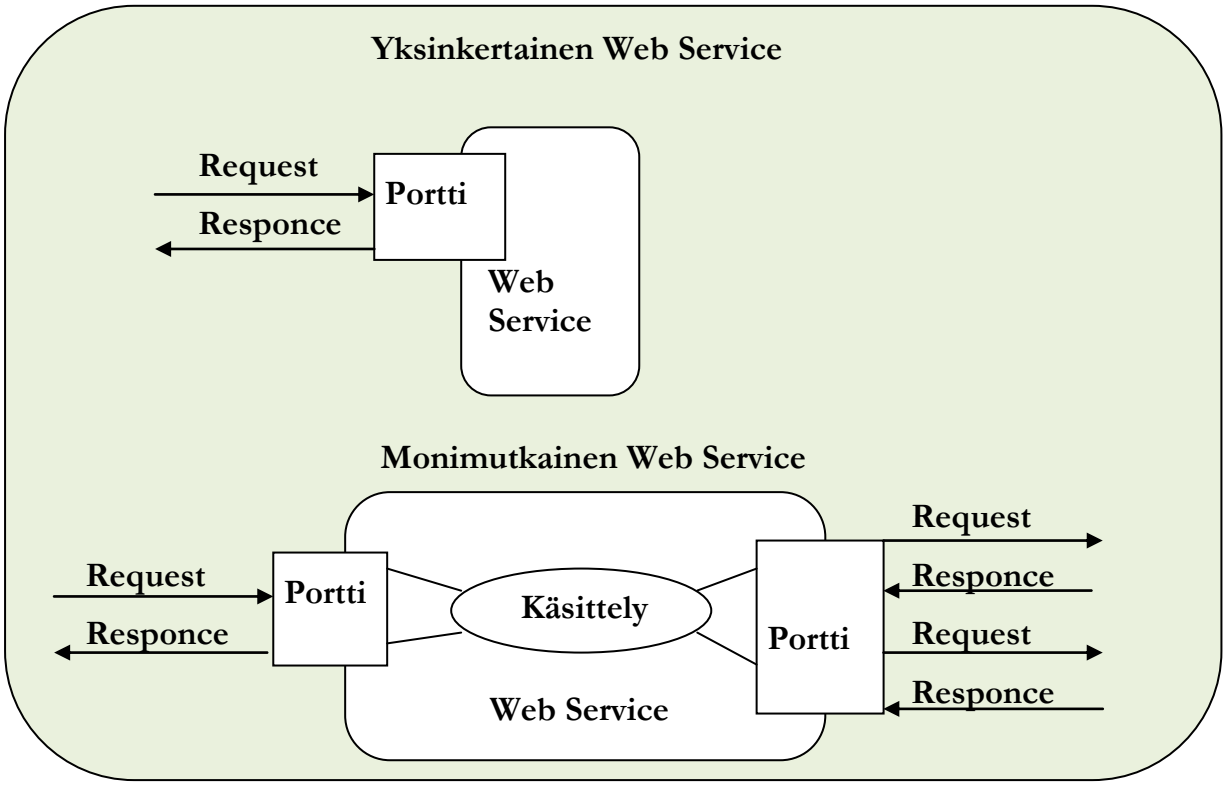
1.3 Käytettävät työkalut ja ohjelmointitekniikat

Projektissa käytettiin itse Web Servicen kehityksessä Oraclen JDeveloper työkalua, versio 10.1.3.5.0. Tämän version käyttö tukee sitä, että valmiin Web Servicen palvelimelle siirto- prosessi sujuisi mahdollisimman kivuttomasti jo olemassa olevalle Oracle Application Server 10g 10.1.3.5.0 palvelimelle. Web Servicen ohjelmointikielenä kehityksessä oli Java. Uudet tietokantaproseduurit autentikaatio- ja auktorisointitietojen tarkistusta varten toteutettiin PL/SQL- ohjelmointikielellä Oraclen tietokannassa.

2 Web Services

Web Service on julkinen ohjelmistorajapinta, jota voidaan kutsua sovelluksen ulkopuolelta. Tämä tarkoittaa sitä, että sen tarjoamaa julkista palvelua voidaan käyttää verkon yli. Herättäminen vaatii sen, että sitä kutsuva sovellus tai toinen Web Service tietää missä osoitteessa ja mitä porttia kutsuttava Web Service kuuntelee. Web Service rajapinnan toteutus voidaan tehdä eri ohjelmointikielillä. Toisiaan kutsuvien Web Service rajapintojen toteutuksessa käytetty ohjelmointikieli ei ole sidonnainen sitä kutsuvan sovelluksen tai toisen Web Servicen toteutuksen ohjelmointikielestä. Tämä johtuu siitä, että kommunikointikieli on aina XML- muotoista. Web Servicen operaatiot, näiden parametrit, sen tukemat protokollat ja tyyli kuvataan Web Servicen WSDL (Web Service Description Language) tiedostossa.(Papazoglou.2012,14–19.)

Web Servicet voidaan jakaa kahteen eri tyyppiin, yksinkertaiseen tai monimutkaiseen. Yksinkertainen Web Service toimii käytännössä Request-Responce tyylisesti käsitellen sitä kutsuneen sovelluksen lähettämän Request- sanoman sisällön. Web Servicessä tehdään Request- kutsun perusteella mahdollisesti eri toimenpiteitä ja palauttaen Responce- sanomassa vastaus Request- sanoman lähettäneelle sovellukselle. RPC-tyylisen Web Servicen herättävässä Client- sovelluksen lähettämässä Request-sanomassa voi olla esimerkiksi funktion kutsu parametreilla, jonka perusteella tehdään käsittely ja palautetaan funktion paluuarvo Responce- sanomassa. Monimutkaisessa Web Servicessä otetaan vastaan Request- sanoma, jonka sisällön perusteella suoritetaan erilaisia prosesseja ja kutsuja toisiin Web Service- rajapintoihin. (Papazoglou.2012, 19–21.)



Kuva 1. Karkean tason kuvaus yksinkertaisesta ja monimutkaisesta Web Servicestä (Papazoglou.2012,20.)

2.1 WSDL

WSDL, eli Web Services Description Language, on nimensä mukaisesti Web Servicen XML- muotoinen kuvaus, jonka tietojen perusteella kyseistä Web Serviceä käyttävät

Client- sovellukset ja toiset Web Servicet osaavat muodostaa oikeanlaisia Request-sanomia Web Serviceen käyttäen Web Servicen tukemia protokollia. (Papazoglou.2012,151–153.)

```
<definitions targetNamespace="http://starmans_example_proj/"
name="StarmansExampleService" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:tns="http://starmans_example_proj/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
```

Kuva 2. Definitions elementin attribuutteina esiteltävät nimiavaruudet.

WSDL:n juurielementti on definitions joka myös esittelee nimiavaruudet, joita kyseisessä WSDL kuvauksessa käytetään. WSDL:n sisällön voi jakaa kahteen osaan: abstraktiin ja fyysiseen kuvaukseen. Abstraktissa kuvauksessa esitellään Web Servicen eri rajapinnat (portType), niiden sisältämät operaatiot (operation), ja näiden operaatioiden kanssakäymisessä käytettävät viestit (message), sekä viestien parametrit (Part) ja niiden tyypit. Javassa rajapintojen kuvaavaa portType elementtiä voisi verrata julkiseen rajapintaan (public interface), operaatiot ja niitä kutsuttavat viestit (incoming message), sekä niiden palauttamat viestit (output message) olisivat verrattavissa tuon kyseisen rajapinnan sisältämiin julkisiin metodeihin, joiden parametreja ovat Part elementit.(Erl.2008. Service-Oriented Architecture, 131–136; Papazoglou.2012,151-1-69.)

```
<message name="sampleMethod">
  <part name="arg0" type="xsd:string"/>
</message>
<message name="sampleMethodResponse">
  <part name="return" type="xsd:string"/>
</message>
<portType name="StarmansExample">
  <operation name="sampleMethod">
    <input wsam:Action="http://starmans_example_proj/StarmansExample/sampleMethodRequest" message="tns:sampleMethod"/>
    <output wsam:Action="http://starmans_example_proj/StarmansExample/sampleMethodResponse" message="tns:sampleMethodResponse"/>
  </operation>
</portType>
```

Kuva 3. Abstrakti kuvaus RPC- tyylin Web Servicen WSDL- tiedostossa.

Kuvassa 3 on malliksi tekemästäni RPC- tyylisen Web Servicen WSDL tiedostosta abstrakti osa, josta näkyy messaget ”sampleMethod” ja ”sampleMethodResponse”. Näillä kummallakin on parametrina merkkijono- tyyppiset parametrit. PortType ”StarmansExample” kuvaa abstraktia porttia, jolla tässä esimerkissä on yksi operaatio, jonka nimi on ”sampleMethod”. Tämä on julkinen metodi ”sampleMethod”, joka ottaa parametrinaan sisäänsä merkkijono tyyppisen parametrin, joka määriteltiin message- elementissä ”sampleMethod”. Metodi palauttaa viestin merkkijono tyyppisenä, joka on määritelty message- elementissä ”sampleMethodResponce”.

Fyysisen kuvauksen (concrete description) elementtejä ovat binding-, service- ja port- elementit. Binding- elementti sitoo nimensä mukaisesti abstraktissa osuudessa esitellyn portType- elementin. Binding- elementissä määritellään portType- elementin fyysinen tiedonsiirtoprotokolla. Tässä esimerkissä tiedonsiirtoprotokollana on HTTP. Binding- elementissä attribuuttina annetaan myös type- attribuutti. Type- attribuutissa kerrotaan, onko SOAP RPC- vai Document- tyylinen. Service- elementti kertoo palvelun nimen jonka sisällä oleva port- elementti kertoo Web Servicen kuuntelevan portin fyysisen osoitteen.(Erl.2008. Service-Oriented Architecture, 131–136;Papazoglou.2012,151-1-69.)

```
<binding name="StarmansExamplePortBinding" type="tns:StarmansExample">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="sampleMethod">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://starmans_example_proj/"/>
    </input>
    <output>
      <soap:body use="literal" namespace="http://starmans_example_proj/"/>
    </output>
  </operation>
</binding>
<service name="StarmansExampleService">
  <port name="StarmansExamplePort" binding="tns:StarmansExamplePortBinding">
    <soap:address location="http://localhost:7101/Application1-Starmans_example_proj-context-root/StarmansExamplePort"/>
  </port>
</service>
```

Kuva 4. Fyysinen kuvaus RPC- tyylin Web Servicen WSDL- tiedostossa.

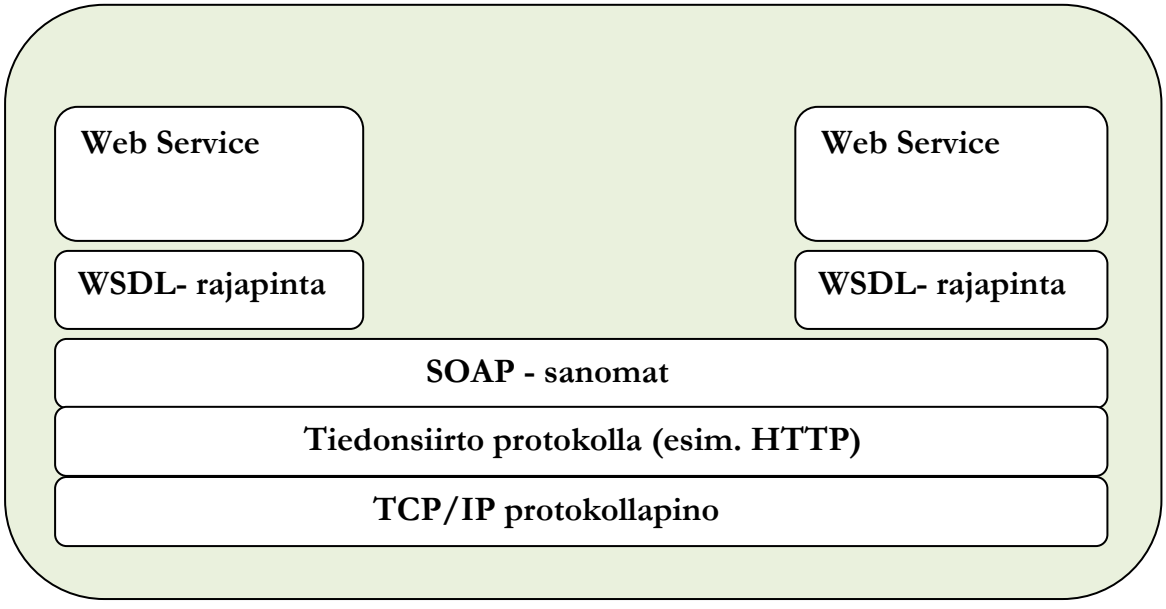
Kuvassa 4 oleva binding- elementti sitoo abstraktin ”StarmansExample” portTypen ja kertoo, että tiedonsiirtoprotokollana on HTTP. Elementissä kerrotaan myös, että kutsut ovat SOAP- kehyksen mukaisia ja RPC- tyylisiä. Operaation ”sampleMethod” SOAP sanoman payload, eli body osio määritellään literal tyyppiseksi. Service elementti

kertoo tämän Web Servicen nimen ”StarmansExampleService”, jossa on yksi portti, jonka osoite kerrotaan soap:address- elementin location- attribuutissa.

Web Serviceä käyttävät Client- sovellukset pystyvät rakentamaan niin sanotun Web Service Stubin koneellisesti WSDL tiedostosta. Tämä tarkoittaa sitä, että Client-sovellus pystyy tekemään rungon Web Servicen kutsua varten. Rungossa on luotuna oikeanlaiset kutsut Web Servicen fyysisiin portteihin sekä metodit, niiden parametrit ja parametrien tietotyypit, jotka on esitelty WSDL- tiedostossa. Stubin oleellisin asia on XML muotoisen SOAP sanoman muuntaminen Client- sovelluksessa toteutettuun ohjelmointikieleen, sekä XML muotoisen SOAP sanoman muodostaminen Client-sovelluksen ohjelmointikielellä tehtyjen metodikutsujen perustella.

2.2 SOAP

SOAP eli Simple Object Access Protocol on XML- muotoinen protokolla, jota käytetään kommunikaatiossa Web Serviceen. Se mahdollistaa yleisiin standardeihin sitoutuneen viestintämuodon tiedonsiirroissa Web Service- rajapintojen välillä, eikä vaadi tietoa itse ohjelmointikielestä, jolla Web Service on toteutettu. SOAP sanomat kulkevat Web Servicen ja sitä kutsuvan sovelluksen tai toisen Web Servicen välillä tiedonsiirtoprotokollien pyyntöjen (Request- sanoma) ja vastausten (Responce-sanoma) sisällä. SOAP sanomien yleisimmin käytetty tiedonsiirtoprotokolla on HTTP. Muita tiedonsiirtoprotokollia joiden avulla SOAP sanomia voidaan kuljettaa Web Servicen ja sitä kutsuvan sovelluksen tai toisen Web Servicen välillä ovat FTP, SMTP ja RMI.(Papazoglou.2012,126–128.)



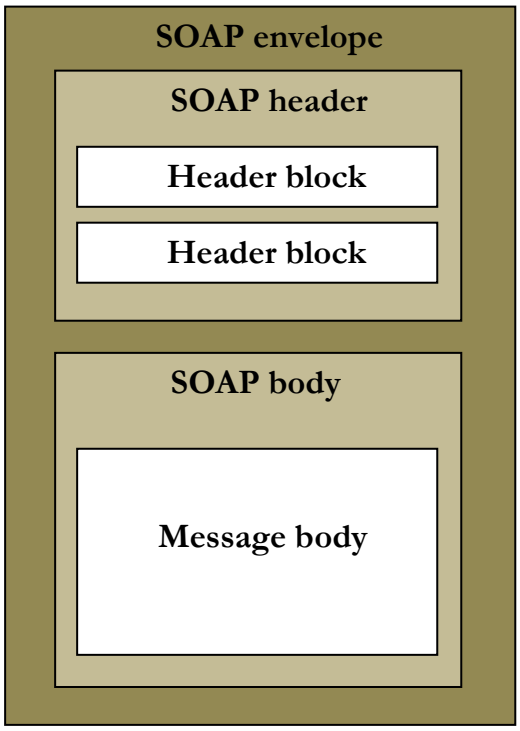
Kuva 5. Web Service kommunikaatio kuvaus(Papazoglou.2012,129.)

Kuva 5 näyttää kuinka Web Service lähettää SOAP sanoman tiedonsiirtoprotokollalla, esim. HTTP- Request sanoman sisältönä TCP- streamissa kohteena olevalle Web Servicelle. Vastaanottajan HTTP- kuuntelija välittää HTTP- Request sanomassa olevan SOAP sanoman SOAP- prosessoijalle, joka prosessoi sanoman sisällön. Prosessoituaan sanoman vastaanottava Web Service palauttaa HTTP Responce- sanomassa vastauksen SOAP sanomana Request- sanoman lähettäneelle Web Servicelle.(Papazoglou.2012,128.)

SOAP- sanoman rakenne koostuu pakollisesta SOAP envelopesta ja sen sisällä olevista mahdollisesta SOAP header osasta, sekä pakollisesta SOAP bodysta. SOAP rakennetta voisi verrata kirjeeseen: SOAP envelope on kirjekuori, jonka käsittelysäännöt kerrotaan SOAP header osassa, jossa voidaan määrittää esimerkiksi kirjeen käsittelyyn ja reittiin liittyviä asioita. (Erl.2008. Service-Oriented Architecture, 142–144.)

SOAP header- elementissä määriteltäviä asioita ovat esimerkiksi autentikointi- ja auktorisointimäärittelyt, tiedon eheyden määrittelyt, alkuperäinen lähettäjä, lopullinen vastaanottaja, sekä välissä SOAP sanomaa prosessoivat vastaanottajat. SOAP header-elementtiin määritelty välivastanottaja prosessoi sille tarkoitetun osan header-elementissä, poistaa sen ja mahdollisesti muokkaa SOAP sanomaa ja lähettää sanoman edelleen eteenpäin. (Erl.2008. Service-Oriented Architecture, 142–144.)

SOAP bodya voitaisiin kuvailla kirjeen sisällä oleva tiedoksi, payloadiksi. SOAP sanomassa body- elementti on pakollinen ja sen sisältönä on itse käsiteltävä sanoma. Jos SOAP sanoman käsittelyssä tulee virheitä, niin virheet ilmoitetaan paluusanomassa lähettäjälle body- elementin sisällä fault- elementissä. Body voi sisältää vain joko sanoman tai fault- elementin, eli nämä eivät voi esiintyä samanaikaisesti body-elementin sisällä.(Erl.2008. Service-Oriented Architecture, 142–144.)



Kuva 6. SOAP sanoman rakenteen kuvaus (Papazoglou.2012,136.)

2.3 RPC ja Document tyylinen Web Service

RPC tyylisessä (Remote Procedure Call) Web Servicessä operaatiot ovat Web Servicen julkisten metodien kutsuja joita kutsutaan niille määritellyillä parametreilla, joihin Web Service vastaa niille määritellyillä paluuarvoilla. Nämä metodi kutsut esitetään SOAP sanomassa XML- elementteinä. RPC tyylisessä Web Servicessä SOAP sanoman XML-muodossa oleva metodikutsu käsitellään ja metodin palauttama paluuarvo tai virhe lähetetään asiakkaalle SOAP- sanomassa XML- muotoisena. RPC- tyylinen Web Service on synkroninen. Tämä tarkoittaa sitä, että kun Web Serviceä kutsutaan HTTP-Request sanomalla, kutsuja on sidottu kutsuun odottamaan HTTP- Responce sanomaa hänen lähettämästään Request sanomasta.(Papazoglou.2012,72–74,139-141.)

Document (message) tyylisessä Web Servicessä kutsun sisältö on Web Servicen WSDL tiedostossa määritellyn scheman mukainen XML- dokumentti, jonka sisältöä ei prosessoida samalla tavalla kuin RCP- tyylisessä Web Servicessä, jossa ennen metodin

herättämistä täytyy metodi kutsun parametrit olla oikeanlaiset.(Papazoglou.2012,142–144.)

3 Tietoturva

Web Service rajapintojen luonteen vuoksi tietoturvasta huolehtiminen Web Servicen toteutuksessa on erittäin tärkeää. Koska Web Servicen tarkoituksena on nimenomaan mahdollistaa verkon yli tapahtuva XML- muotoinen tietojen siirto, aiheuttaa tämä riskejä. Yksi riski on autentikointi ja auktorisointi, eli onko Web Serviceä kutsuva käyttäjä oikeasti se joka väittää olevansa ja onko hänellä oikeus käyttää Web Serviceä. Toinen riski on tiedon muuttumattomuus, eli onko käyttäjän lähettämä sanoma oikeasti se, jonka hän on lähettänyt vai onko joku muu taho muokannut sanomaa käyttäjän ja Web Servicen välillä.(Papazoglou.2012,416–420.)

Tässä projektissa tilaajan vaatimuksena oli projektin osalta tietoturvaratkaisuna käyttää auktorisoinnin ja autentikoinnin osalta OASIS WS-Security UsernameToken standardia. Tiedon muuttumattomuuden turvaamista ei toimeksiantajan vaatimusmäärittelyssä vaadittu ja sitä ei toteutettu tässä projektissa. Tästä syystä opinnäytetyössä käsitellään vain toteutettuun ratkaisuun liittyvää tietoturvaratkaisua, WS-Security UsernameToken Profilea.

3.1.1 Custom JAAS Login Module

Autentikaatio- ja auktorisointitiedot tallennetaan Oracle palvelimella oletuksena XML-muotoisena palvelimen system-jazn-data.xml konfiguraatio tiedostoon, joita vasten palvelin validoi annetut käyttäjä- ja salasanatiedot. Jos autentikaatio- ja auktorisointitiedot halutaan validoida erikseen määritellystä paikasta, kuten tässä projektissa, tarvitaan toteutukseen erillinen Login Module. Palvelimen system-jazn-data.xml tiedostoon täytyy tällaisessa tapauksessa määritellä, että Web Service käyttää autentikaatio- ja auktorisointi tietojen validointiin tätä Login Modulea.(Oracle Custom Login Modules.)

Login Modulessa käsitellään Callbackejä, jotka ovat Web Serviceä kutsuvan käyttäjän lähettämästä kutsusta saatavia tietoja. Näitä ovat Login Modulessa käyttäjätunnus- ja salasanatietoihin liittyen UsernameCallback, sekä PasswordCallback funktiot.(Callback interface.)

Login Modulessa on myös tuki NonceCallback-, DigestCallback- sekä CreatedCallback- funktioille, joita tarvitaan toteuttamassani ratkaisussa, koska lähettävän Client sovelluksen salasana ei tule selkokielisenä salasanana password Callback funktiosta. Näiden Callback funktioiden avulla saadaan UsernameTokenista vaadittavat elementtien nonce, PasswordDigest ja created, tietosisällöt.

3.2 OASIS UsernameTokenProfile

Web Servicen autentikaation varmistamiseksi käyttäjätunnuksen ja salasanan välittäminen Client- sovelluksesta Web Serviceen on oleellista toteuttaa yleisesti hyväksyttyjen ja tunnettujen standardien mukaisesti. OASIS organisaation tarjoamia tapoja pidetään turvallisina tapoina toteuttaa tietoturvallisia autentikointi ja auktorisointi tapoja Web Service- rajapintoihin. Tässä projektissa olen toteuttanut autentikointi ja auktorisointitietojen välittämisen SOAP sanomassa OASIS organisaation määrittelemää UsernameTokenProfile- standardia käyttäen.

Usernametoken- elementissä on pakollisena attribuuttina MustUnderstand, joka määrittelee, että Web Servicen täytyy ymmärtää Usernametoken. Usernametokenin ainoa pakollinen alielementti on username, joka ei yksinään tarjoa minkäänlaista suojaa autentikointitiedon välittämiseen Usernametoken- elementissä. Usernametokenin sallitut alielementit ovat username, password, nonce ja created. Password eli salasana elementti voi olla joko PasswordText tai PasswordDigest tyyppisenä. (Oasis Usernametoken profile.)

PasswordText, eli selkokielinen salasana on nimensä mukaisesti salaamaton selkokielinen salasana. PasswordDigest on SHA1 salausalgoritmilla laskettu base64 dekoodattu arvo, joka muodostuu salasanasta (password), luontiajasta(created), sekä satunaisesta merkkijonosta(nonce) lasketusta algoritmista. PasswordDigestin

muodostus: Base64(SHA-1(nonce + created + password)). Jos Usernametokenin sisältämä password tai PasswordDigestin sisältämä salasana on selkokielinen, vaatii salasanan tarkistus sitä käsittelevältä Web Serviceltä pääsyn selkokielisiin salasanoihin. (Oasis Usernametoken profile.)

Created elementtiä voidaan käyttää hyödyksi PasswordDigestin muodostuksen lisäksi määrittämällä aikaikkuna, jonka sisällä sanoma pitää olla luotu alkuperäiseltä lähettäjältä Web Serviceen. Suosituksena vanhenemisajaksi OASIS määrittää 5 minuuttia. Tämä tarkoittaa sitä, että jos created- elementin aikaleima on vanhempi kuin määritelty vanhenemisaika, sanoma hylätään ja lähettäjälle palautetaan virhe. (Oasis Usernametoken profile.)

Nonce- elementtiä voidaan käyttää hyödyksi PasswordDigestin muodostuksen lisäksi estämään toistohyökkäyksiä (replay attacks). Nonce- elementin sisältö ei saa olla sama vaan se pitää vaihtua. Toistohyökkäyksien estämiseksi Web Serviceen kohdistuneista sanomista säilytetään nonce- elementin arvot tietyn määritellyn ajan, esimerkiksi tuon edellisessä kappaleessa mainitun aikaikkunan ajan Web Servicen muistissa. Web Serviceen saapuvien sanomien nonce- elementtejä verrataan tallennettuihin arvoihin. Sanomat, joiden sisältämä nonce- elementti on Web Servicen muistissa, hylätään. (Oasis Usernametoken profile.)

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu= "...">
   <S11:Header>
      ...
      <wsse:Security>
         <wsse:UsernameToken>
            <wsse:Username>NNK</wsse:Username>
            <wsse:Password Type="...#PasswordDigest">
               weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==
            </wsse:Password>
            <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
            <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
      ...
   </S11:Header>
   ...
</S11:Envelope>
```

Kuva 7. Esimerkki Usernametoken (Oasis UsernameToken profile.)

4 Tehdyt ratkaisut ja perustelut

Finnvallin Web Servicen määrittelyvaiheessa vaatimuksena oli toimeksiantajan osalta se, että tietokantaan tuotavaa aineistoa ei validoida toteutettavassa Web Servicessä. Itse aineiston validointi tapahtuu jo olemassa olevilla tietokantaproseduureilla. Myöskään sisään luettavan aineiston Schemaa, eli XML- muotoisen aineiston kuvausta ei haluttu implementoida Web Serviceen.

Näistä syistä Web Servicen toteutuksessa päädyttiin RPC- tyyliseen toteutukseen, jossa sisään luettava aineisto tuodaan, ja vastaavasti noudettava aineisto haetaan, liitteenä toteuttaen SWA (SOAP With Attachments) tekniikkaa. Tätä käytetään yleisesti silloin kun SOAP- sanomissa kuljetetaan tietoa, joka ei ole XML- muodossa, kuten kuvia. Tämä sopii toteutettavan Web Servicen toteutukseen siksi, koska tällä tavalla SOAP sanoman body osan XML- muotoista aineistoa ei tarvitse validoida Web Servicessä Fivaldi- sovelluksen XML- aineistojen kuvaavaa Schemaa vasten.

4.1 Aineistojen validointi

Toteutuksessa aineistojen validointi Web Servicessä rajoitettiin SOAP sanoman validointiin. Tällä tarkoitan sitä, että validoinnissa tarkistetaan, että sanoma on oikein muodostettu Web Servicen WSDL- tiedoston mukaisesti. WSDL- tiedostossa on määritelty operaatiot joilla toteutettua Web Serviceä voi kutsua. Näiden operaatioiden kutsumiseen määritellyt parametrit esitellään myös WSDL- tiedostossa. Web Servicessä myös näiden parametrien sisältämä tieto validoidaan, tarkoittaen sitä, että tietosisällön täytyy olla sellaista, jolla aineistoja voidaan noutaa sovelluksen tietokannasta ja tuoda sovelluksen tietokantaan.

Web Servicen autentikointi ja auktorisointi toteutetaan OASIS WS-Security UsernameToken- standardilla, jonka käsittelystä kerrotaan tarkemmin kappaleessa 4.4, joka kertoo Web Servicen autentikoinnin ja authorisoinnin toteutuksesta.

4.2 Web Service- rajapintaan toteutetut operaatiot

Web Serviceen toteutettiin tämän projektin osalta kaksi operaatiota, joilla Web Service-rajapintaa voidaan kutsua. Ensimmäistä käytetään XML-muotoisen tiedon sisään tuomiseen sovelluksen tietokantaan. Tämän operaation nimi on addFile. Toista operaatiota käytetään XML-muotoisen tiedon noutamiseen sovelluksen tietokannasta. Tämän operaation nimi on getFile. Näillä operaatioilla on kolme yhtenäistä parametria, jotka ovat schema, businessId sekä parameterObject.

Schema- parametrin sisältämä tieto tarkoittaa tietokantaryhmää, josta aineistoa noudetaan, tai johon aineistoa tuodaan. Tämä tieto tarvitaan, koska tällä hetkellä Finnvallin kehittämän Fivaldi- sovelluksen käyttäjiä on kuudessa eri kantaryhmässä ja aineistojen siirrossa täytyy olla tiedossa missä kantaryhmässä yritys, jonka aineistoja noudetaan tai johon aineistoja tuodaan, sijaitsee. Tämä tieto välitetään Finnvallin kautta Web Servicen käyttäjille erikseen, koska heillä ei ole olemassa olevaa tietoa, missä kantaryhmässä eri yritykset ja niihin liittyvät tiedot sijaitsevat tietokannassa.

BusinessId- parametrin sisältämä tieto tarkoittaa sitä, minkä yrityksen aineistoa noudetaan tai johon aineistoa tuodaan. Aineistojen siirto sallitaan yrityskohtaisesti, koska jatkokehityksessä aineistojen siirtoa tullaan rajaamaan yrityskohtaisesti määrä-, sekä aikarajoitteisesti Web Servicen ja tietokannan kuormituksen tasaamiseksi, sekä yritysten maksamien käyttömaksujen mukaisesti.

ParameterObject- parametri on erikseen räätälöity parametri Web Serviceä varten, joka pitää sisällään listan parametriobjekteja.

GetFile funktion parametreina ovat schema, businessId ja lista parametriobjekteista, ParameterObject. Tuota parametriolioita sisältävää listaa varten jouduttiin toteutukseen tekemään oma Java- luokka. RPC- tyylisessä Web Servicessä SOAP prosessoija muuntaa XML- muotoiset parametrit Web Servicen toteutuksen ohjelmointikielen mukaisiksi. Tässä projektissa ohjelmointikielenä on Java, ja näin ollen prosessoija muuntaa XML- muotoisen SOAP- sanoman vastaamaan toteuttamani Web Servicen kutsuttavia Java- ohjelmointikielellä toteutettuja metodeita ja niiden parametreja.

Tarkempi kuvaus räätälöidyn parametrin käytöstä Web Servicessä on kappaleessa 4.3: Objektien käyttö parametrina Web Servicessä.

GetFile operaation paluuviestin tyyppinä on SOAP AttachmentPart. Tämä tarkoittaa sitä, että Web Servicen vastaussanoma, eli Responce- sanoma käyttäjän lähettämään Web Servicen herättäneeseen Request- sanomaan on SOAP- sanoma, jossa liitteenä on halutuilla parametreilla pyydetty XML-muotoinen aineisto. Jos käyttäjän lähettämän Request- sanoman käsittelyssä ilmaantuu tekninen virhe, tai sanoma ei ole validi, niin Web Servicen käyttäjälle lähetettävässä Responce- vastaussanomassa palautetaan SOAP- sanoman body- elementissä soapFault- elementti. SoapFault- elementti sisältää faultCode- elementin sekä faultString- elementin, joka sisältää selkokielisen selityksen virhetilanteeseen.

AddFile funktion parametreina on schema, businessId, lista parametriobjekteista sekä AttachmentPart, eli SOAP sanoman liite. AttachmentPart sisältää sovelluksen tietokantaan kyseisessä kantaryhmässä olevalle yritykselle parametrien mukaiset aineistot. Operaation herättäneeseen käyttäjän lähettämään Request- sanomaan vastataan Responce- sanomalla. Web Servicen lähettämän Responce- sanoman sisältönä on onnistuneessa aineiston sisään luvussa merkkijonotyyppinen paluuviesti, jossa viestin sisältö on paluukoodi 00 sekä onnistuneesti sisään luetun aineiston koko. Esimerkkinä paluuviesti: 00 12750. Jos Request- sanoman käsittelyssä ilmaantui tekninen virhe, tai sanoma ei ollut validi, niin Responce- sanomassa palautetaan body-elementissä soapFault- elementti. SoapFault- elementti sisältää faultCode- elementin sekä faultString- elementin, joka sisältää selkokielisen selityksen virhetilanteeseen.

4.3 Objektien käyttö parametrina Web Servicessä

Yleisien eri tietotyyppien, kuten merkkijonojen, numeroiden tai päivämäärän muuntaminen Web Servicessä XML- muodosta Web Servicen toteutuksessa käytettyyn ohjelmointikieleen ja päinvastoin tapahtuu automaattisesti SOAP prosessoijan toimesta. Silloin jos parametrina on itse määritelty objekti, tässä tapauksessa parametri-luokka, ei prosessoija osaa automaattisesti kääntää sen sisältöä XML- muodosta Web Servicen toteutuksessa käytettyyn ohjelmointikieleen. Tällaisessa tapauksessa

kääntäminen pitää toteuttaa manuaalisesti. Tässä projektissa kääntäminen toteutettiin XML- muotoisen tiedon ja Java- luokan välillä.

Jotta luokka voitiin integroida kummankin operaation parametriksi, piti tätä varten toteuttaa myös oma Java- luokka ”ParameterObjectSerializer” muuntamista varten, jossa määritellään mikä SOAP- elementin XML- muotoinen kenttä vastaa mitä Parameter- luokan muuttujaa. Luokan tarkoitus on muuntaa XML- muotoisen SOAP-sanoman elementteinä olevat parametrit Java- olioiksi ja Java- oliot XML- muotoisiksi SOAP- sanoman elementeiksi. Tätä varten Web Serviceä kuvaavaan WSDL-tiedostoon piti myös lisätä Schema- kuvaus, jossa kuvataan käytettävän Java- olion sisältämät tietotyypit XML- muodossa. WSDL- tiedostossa olevan Schema- kuvauksen avulla Web Serviceä käyttävä sovellus osaa luoda oikeanlaisen Stubin eli Web Servicen kutsua varten tarvittavan rungon. Stubin avulla Web Serviceä kutsuva sovellus osaa kutsua toteuttamaani Web Serviceä oikeanlaisilla parametreilla myös tämän parametri olion osalta.

```
private String getTextContent(SOAPElement element) {
    String str = "";
    for (Iterator i = element.getChildElements(); i.hasNext();) {
        Object obj = i.next();
        if (obj instanceof Text) {
            str += (((Text) obj).getValue());
        }
    }
    return str;
}

public Object deserialize(SOAPElement element) {
    ParameterObject object = new ParameterObject();
    String parameter = null;
    Date fromDate = new Date();
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'hh:mm:ss");
    DateFormat formatter ;
    formatter = new SimpleDateFormat();
    for (Iterator i = element.getChildElements(); i.hasNext();) {
        Object obj = i.next();
        if (obj instanceof SOAPElement) {
            SOAPElement child = (SOAPElement)obj;
            if (child.getLocalName().equals("parameter")) {
                System.out.println(getTextContent(child));
                parameter = (getTextContent(child));
                object.setParameter(parameter);
            }
            if (child.getLocalName().equals("fromDate")) {
                try {

                    System.out.println(getTextContent(child));
                    fromDate = df.parse(getTextContent(child));
                    object.setFromDate(fromDate);

                } catch (ParseException e) {
                    System.out.println("dateformat not valid");
                    e.printStackTrace();
                }

            }
        }
    }
    return object;
}
```

Kuva 8. XML- muotoisen parametri- objektin muuntaminen Java- olioksi toteutetussa ParameterObjectSerializer Java- luokassa.

4.4 Web Service- rajapintaan toteutettu tietoturvaratkaisu

Tässä projektissa tietoturvaratkaisuna toteutettiin autentikointi ja auktorisointitietojen välitykseen SOAP- sanomassa OASIS- organisaation määrittelemän WS-Security-

standardin mukaisen UsernameToken- ratkaisu, jossa vaatimuksena on salasanatiedon salaaminen algoritmilla. Tämä tarkoittaa sitä, että salasana on PasswordDigest-tyyppinen. PasswordDigest- elementin tieto muodostetaan seuraavasti: PasswordDigest = Base64(SHA-1(nonce + created + password)).

Vaatimuksena lähettävältä osapuolelta on muodostaa toteutettuun Web Serviceen SOAP- sanoman header- osaan WS-Security- standardin mukainen UsernameToken autentikointi ja auktorisointi. UsernameToken- elementin alielementteinä vaadittavat elementit ovat username, nonce, created sekä password, joka täytyy olla PasswordDigest- tyyppinen. Vaatimuksena projektissa oli se, ettei salasanoja pidetä tallessa ja ylläpidetä palvelimella jossa Web Service sijaitsee. Tästä syystä autentikoinnin ja auktorisoinnin varmistamiseen ei voitu käyttää Oraclen OC4J- palvelimen oletuksena olevaa system-jazn-data.xml- konfiguraatiotiedostoa.

4.5 JAAS Login Module

Jotta autentikointi ja auktorisointi tässä projektissa voitiin toteuttaa tietokannasta, ei normaali autentikointi- ja auktorisointitietojen oikeellisuuden tarkistus edellä mainitusta konfiguraatiotiedostosta tullut kysymykseen. Web Serviceen toteutettiin JAAS Login Module. Normaalisti tällaisessa erikseen tehdyssä Login Modulessa suoritetaan autentikointi ja auktorisointi, ja näiden onnistuessa sanoma käsitellään. Jos autentikointi tai auktorisointi epäonnistuu, sanomaa ei käsitellä. Silloin Request- sanomaan vastataan Responce- sanomalla, joka sisältää virhe- elementin autentikaation ja auktorisoinnin epäonnistumisesta. Jos projektin määrittelyssä olisi riittänyt vaatimuksena, että autentikointi ja auktorisointi rajattaisiin vain itse Web Servicen käyttöön, olisi logiikka toteutettu edellä mainitulla tavalla.

Projektin määrittelyssä kuitenkin haluttiin rajata käyttäjätunnuksen oikeuksia eri aineistotyyppeihin, joita voidaan noutaa ja tuoda sovellukseen, joten pelkkä auktorisointi Web Servicen sekä operaatioiden käyttöön ei ole riittävän tarkka. Tällä tarkoitan sitä, että autentikoinnin läpäisevällä käyttäjätunnuksella ja salasanalla voi olla määritelty oikeudet Web Servicen käyttöön, mutta aineistosiirtojen osalta oikeudet vain johonkin tiettyyn aineistoon tietyltä yritykseltä. Tässä projektissa haluttiin auktorisointi

toteuttaa tarkemmalla tasolla, toisin sanoen tarkistaa samalla mitä operaatiota autentikoitu käyttäjä kutsuu ja millä parametreilla.

Otetaan esimerkkinä käyttäjätunnus, jolle on määritelty oikeudet Web Servicen käyttöön, sekä esimerkiksi vain tuoterekisteri- aineiston noutoon tietyltä asiakkaalta sovelluksessa. Jos käyttäjä haluaa noutaa sekä tuoterekisteri-, että yritysrekisteriaineiston, niin auktorisoinnin tuloksena on pyynnön epääminen. Tämä johtuu siitä, että hänelle ei ole määritelty oikeuksia yritysrekisteriaineiston noutoon.

Toteutetussa Login Modulessa ei päästä käsiksi suoraan sanoman sisältöön, eli juuri näihin edellisessä kappaleessa mainittuihin auktorisoinnin kannalta oleellisiin tietoihin, joten ratkaisuna päädyttiin toteuttamaan vaadittava auktorisointi seuraavasti:
Login Modulessa tarkistetaan vain, että itse UsernameToken on muodostettu oikein ja sisältää vaadittavat elementit. Jos näin ei ole, niin autentikointi ja auktorisointi epäonnistuu Login Modulessa ja lähettäjälle palautetaan Responce- sanomassa SOAP-sanoma, joka sisältää WS-Security- standardin mukaisen virheen autentikoinnin ja auktorisoinnin epäonnistumisesta. Jos itse UsernameToken on muodostettu oikein, läpäisee sanoma tässä vaiheessa autentikoinnin ja auktorisoinnin. Itse varsinainen autentikointi ja auktorisointi toteutetaan tietokantaproseduurissa, johon viedään kaikki vaadittavat tiedot ennen operaation kutsua toteutetussa messageHandler Java- luokassa. Tätä toimintoa varten Login Modulessa otetaan valmiiksi määritellyillä Callback-funktioilla UsernameTokenista tarvittavat username-, nonce-, created- ja digest-elementtien tietosisällöt. Nämä tiedot viedään parametreina messageHandler Java-luokalle.

4.6 MessageHandler Java- luokka

Sanoman validointia ja virhe- elementin muodostusta varten toteutettiin Web Serviceen messageHandler Java- luokan joka implementoi Java- ohjelmointikielen valmista Handler- rajapintaluokkaa. Handler- rajapintaluokka sisältää valmiita metodeja, joiden avulla pääsee käsittelemään ja muokkaamaan SOAP- sanoman sisältöä (Java Handler.)

Toteutettu messageHandler Java- luokka käy läpi koko Web Servicen herättämän Request- sanoman sisältämän SOAP sanoman sisällön. Luokan handleRequest-metodiin toteutettiin logiikka, jossa käydään läpi koko Request- sanoman sisältämä SOAP- sanoma. Tässä metodissa tarkistetaan, että SOAP- sanoman sisältö on oikein muodostettu ja elementtien tietosisältö on validia ennen varsinaisen operaation määrittelemää metodikutsua. SOAP- sanomaa läpikäydessä muodostetaan Java-ohjelmointikielellä XML- dokumentti kutsutusta operaatiosta ja sen parametreista auktorisointia varten, joka sisältää kutsutun operaation ja sen parametrit. Jos messageHandler Java- luokka havaitsee virheellisiä parametreja tai väärin muodostetun SOAP- sanoman, sanoman käsittely lopetetaan ennen operaation määrittelemää metodikutsua. Tässä tilanteessa muodostetaan SOAP- sanomaan soapFault- elementti selkokielisen selityksen kanssa ja lähetetään käyttäjälle takaisin Responce- sanoma, joka sisältää tuon messageHandler Java- luokan luoman soapFault- elementin. SoapFault-elementin alielementtiin, faultString- elementtiin, lisättiin tarkempi kuvaus käyttäjälle virheeseen johtaneesta tilanteesta. Kyseiseen elementtiin lisättiin tietosisällöksi virheen tyypin perusteella selkokielinen ilmoitus, jossa kerrotaan, mikä elementti on virheellisesti muodostettu tai jonka tietosisältö ei ole validi. Tässä tapauksessa sanomaa ei käsitellä enempää eikä autentikointi- ja auktorisointitietoja käsitellä.

Jos SOAP- sanoman rakenne on oikein muodostettu, suoritetaan auktorisointi ja auktorisointi tietokannassa UsernameTokenista saatujen parametrien, sekä SOAP-sanomasta muodostetun XML- dokumentin avulla. Itse autentikointi- ja auktorisointitiedot lähetetään tietokantaan autentikoinnin ja auktorisoinnin toteuttavalle tietokantaproseduurille. Proseduuria kutsutaan messageHandler Java- luokasta HTTP-Request sanomalla, jonka Responce- sanomassa tulee autentikoinnin sekä auktorisoinnin onnistuessa paluuviestinä Web Serviceen ”00 ok”. Tässä tapauksessa SOAP- sanoma läpäisee MessageHandler Java- luokan ja Web Servicen operaation kutsuun määritelty Java- metodi suoritetaan. Itse operaatioiden toiminnan suorittavat metodit toteutettiin Service Java- luokkaan. Nämä metodien nimet ovat samat kuin operaatiot, eli addFile ja getFile. Jos autentikointi tai auktorisointi epäonnistuu, palautuu kutsutusta proseduurista Responce- sanomassa virhekoodi ja epäonnistumisen syy. Esimerkkinä: ”40 ei oikeuksia aineistoon(tuoterekisteri)”.

Jos tällainen autentikoinnin tai auktorisoinnin epäonnistumisen virhe tapahtuu, ei sanomaa käsitellä enempää ja käyttäjän lähettämään Request- sanomaan vastataan Responce- sanomalla. Tuohon palautettavan Responce- sanoman sisältämään SOAP-sanomaan muodostetaan messageHandler Java- luokassa WS-Security- standardin mukainen virhe- elementti, johon lisätään vielä tuon proseduurin palauttama tarkempi virheteksti.

4.7 Autentikointi ja auktorisointi PL/SQL- tietokantapaketissa

Autentikoinnin ja auktorisoinnin toteutus haluttiin projektin määrittelyssä suorittaa sovelluksen tietokannassa. Vaatimus toteutettiin seuraavalla tavalla: Tietokantaan tehtiin PL/SQL- tietokantapaketti (PL/SQL- Package), johon toteutettiin tietokantaproseduuri (Procedure). Proseduurissa toteutetaan autentikointi ja auktorisointi Web Servicen, operaatioiden sekä haluttujen aineistojen tarkkuudella. Kuten edellisessä kappaleessa mainitsin, proseduuria kutsutaan messageHandler Java-luokasta HTTP Request- sanomalla. Proseduurille annetaan kutsussa parametreina seuraavat parametrit: i_username, i_nonce, i_password_digest, i_created sekä i_parameters. Näistä i_parameters- parametri sisältää messageHandler Java- luokassa muodostetun XML- dokumentin, eli annetun operaation, sekä sen sisältämät parametri oliot XML- muotoisena. Parametrit halutaan XML-muotoisena jatkokehitystä ajatellen, jolloin proseduuria kutsuttaessa ei jokaista haluttua aineiston noutoon tai hakuun tarvittavaa parametria tarvitse proseduurin esittelyssä erikseen mainita. Jatkokehityksessä uusien toimintojen parametrien käsittelyyn riittää tuon i_parameters parametrin sisältämän XML-muotoisen sisällön tarkistamiseen toteutetun logiikan muutos uusien toimintojen osalta.

Parametrien nimeämisessä logiikkana on se, että PL/SQL – tietokanta proseduureissa, sekä funktioissa on hyvä käyttää proseduurin tai funktion sisään tulevissa parametreissa edessä i-kirjainta (in) ja ulos annettavissa parametreissa edessä o-kirjainta (out). Tämä johtuu siitä, että kehittäjä pystyy seuraamaan koodia paremmin pidemmissä proseduureissa näkemällä parametrin nimen alusta, onko parametri sisään tullut vai ulos menevä parametri.

Proseduurissa ensimmäinen vaihe on autentikointi, jossa tarkistetaan onko Web Service kutsun tehnyt käyttäjä olemassa tietokannassa. Autentikointi suoritetaan kaksiosaisesti: Ensin tarkistetaan onko tietokannassa olemassa käyttäjä annetulla i_username-käyttäjätunnuksella hakemalla kyseisen käyttäjätunnuksen salasana. Tämän projektin osalta rajattiin pois uusien Web Serviceä varten toteutettavien kumppanikäyttäjätunnusten lisäys tietokantaan, johtuen siitä, että nämä tullaan toteuttamaan eri tavalla kuin nykyisten käyttäjien tietojen ja oikeuksien toteutus. Tämän Web Servicen toteutusta varten lisättiin tietokantaan oma käyttäjätunnus, jonka selkokielinen salasana oli minun tiedossani. Olemassa olevat salasanat ovat salattu MD5- salauksella, joten käyttäjän lähettämän PasswordDigestin muodostamiseen käytetyn selkokielisen salasanan tarkistusta varten täytyy autentikaation tarkistamista varten olla tiedossa selkokielinen salasana.

Tämän projektin jatkokehityksessä toteutettavien kumppanikäyttäjätunnusten salasanat tullaan tallentamaan tietokantaan käyttäen AES- salausta symmetrisellä salausavaimella, joka tallennetaan käärittyyn (Wrapped PL/SQL Package) PL/SQL- tietokantapakettiin. Tämän perusteella tietokantaproseduurissa jatkossa voidaan purkaa tietokannasta haettu käyttäjätunnuksen salasana selkokieliseksi, jota voidaan verrata käyttäjältä saadun PasswordDigestin muodostamiseen käytettyyn selkokieliseen salasanaan.

Autentikoinnin toisessa vaiheessa tarkistetaan salasanan täsmääminen PasswordDigestin sisältämään salasanaan. Tätä varten toteutettiin erillinen Java-luokka, jossa tuo vertaaminen tehdään muodostamalla UsernameToken ja verrataan sen PasswordDigest- elementtiä käyttäjältä saadun UsernameTokenin PasswordDigest-elementtiin. Tämä PasswordDigest- elementin muodostus tehdään käyttämällä käyttäjältä saatuja i_nonce-, i_username- ja i_created- parametreja, sekä sovelluksen tietokannasta saatua salasanaa. UsernameTokenin ja sen sisältämän PasswordDigestin muodostaminen on huomattavasti helpompi toteuttaa Java-ohjelmointikielellä kuin PL/SQL- ohjelmointikielellä, joten sen toteuttaminen Javalla oli nopeampi ja selkeämpi ratkaisu. Toteutettu Java- luokka sisältää funktion, joka palauttaa Boolean arvon, eli tosi tai epätosi arvon. Jos autentikointi tiedot täsmäävät, palautuu arvo tosi ja jos ei, arvo epätosi.

Jos arvo on epätosi, ei auktorisointia tehdä, koska annetun käyttäjätunnuksen salasana ei täsmää tietokannassa olevaan salasanaan. Tässä tapauksessa proseduuri palauttaa messageHandler Java- luokasta lähetettyyn Request- sanomaan Responce- sanoman, jonka sisältö on virhekoodi. Tämän perusteella messageHandler Java- luokassa muodostetaan Web Serviceä kutsuneen käyttäjän lähettämään Request- sanomaan vastauksena Responce- sanoma, jossa sisältönä on SOAP- sanoma sisältäen WS-Security- standardin mukaisen virhe- elementin autentikoinnin ja auktorisoinnin epäonnistumisesta. Jos taas arvo on tosi, niin proseduurissa suoritetaan seuraavaksi auktorisointi i_parameters- parametrin sisältämän XML- muotoisen tiedon perusteella.

Jotta Oraclen tietokannassa voidaan käyttää Java- luokkaa, täytyy tuo Java- luokka ladata ja kääntää tietokantaan, jolloin sitä voidaan käyttää funktiona ja kutsua kuten PL/SQL-funktiota lisäämällä funktion esittelyyn tieto siitä, että kyseessä on ohjelmointikielenä Java(Java Oraclen tietokannassa.)

Esimerkkinä normaali funktion esittely PL/SQL-paketissa:

”FUNCTION NORMAL_FUNCTION (I_YOUR_VARIABLE) RETURN VAR-CHAR2;”

Java- ohjelmointikielellä olevan funktion esittely PL/SQL-paketissa:

”FUNCTION JAVA_FUNCTION IN LANGUAGE JAVA (I_YOUR_VARIABLE IN VARCHAR2) RETURN VARCHAR2;”

Auktorisointi vaiheessa proseduurissa käydään XML- tyyppinen i_parameters-parametrin sisältö läpi PL/SQL ohjelmointikielen valmiin XML- aineiston läpikäymiseen suunnitellun kirjaston avulla(Oracle XML Type Operations.)

Auktorisointi suoritetaan jokaisen eri aineiston kohdalla operaation, käyttäjätunnuksen, kantaryhmän, yrityksen ja aineiston tarkkuudella. Jos autentikoidulla käyttäjällä on oikeudet operaatioon kaikkien haluttujen annetussa kantaryhmässä olevan yrityksen

aineistojen osalta, onnistuu auktorisointi. Tässä tapauksessa proseduuri palauttaa messageHandler Java- luokasta lähetettyyn Request- sanomaan Responce- sanoman, jonka sisältö on autentikoinnin ja authorisoinnin onnistumisesta kertova koodi ”00 ok”. Jos yhdenkin pyydettävän tai tuotavan aineiston kohdalla auktorisointi epäonnistuu niin proseduuri palauttaa messageHandler Java- luokasta lähetettyyn Request- sanomaan Responce- sanoman, jonka sisältö on virhekoodi ”40” sekä selkokielinen erittely aineistoista joihin käyttäjätunnuksella ei ole oikeuksia. Esimerkkinä paluusanoma: ”40 Ei oikeuksia aineistoon (yritysrekisteri)”. Tämän virhekoodin perusteella messageHandler Java- luokassa muodostetaan Web Serviceä kutsuneen käyttäjän lähettämään Request- sanomaan vastauksena Responce- sanoma. Sanoman jossa sisältönä on SOAP- sanoma sisältäen WS-Security- standardin mukaisen virhe- elementin autentikoinnin ja authorisoinnin epäonnistumisesta, sekä tarkemman kuvauksen eri aineistoista, joihin käyttäjätunnuksella ei ole oikeuksia.

4.8 Service Java- luokka

Jos autentikointi ja auktorisointi on onnistunut, voidaan varsinainen metodikutsu suorittaa. Operaatiosta riippuen suoritetaan joko addFile- tai getFile- metodi, jotka ovat tehty Service Java- luokkaan. Tässä vaiheessa myös SOAP- sanoman rakenne ja sen sisältämä tieto on käyty läpi messageHandler Java- luokassa, joten parametreja ei validoida erikseen enää tässä vaiheessa. Kummassakin metodissa käytetään aineiston tuomisessa sekä hakemisessa HTTP Request- kutsuja valmiiksi olemassa oleviin tietokantaproseduureihin. Aineiston hakemiseen tarkoitettuun proseduuriin lähetettävään Request- kutsuun saadaan metodin kutsussa saaduista parametreista suoraan otettua parametreiksi schema, sekä businessId. Tämän projektin osalta, koska Web Serviceen jatkokehityksessä tulevat eri käyttäjätunnukset ja niiden oikeuksien hallinta Web Servicen ja aineistojen siirtojen kanssa ei ole toteutettu, viedään ennalta määritelty käyttäjätunnus ja salasana parametreiksi myös tietokantaproseduuriin lähetettävään kutsuun. Tällä hetkellä getFile- operaatiossa käytettävä aineiston noutoa varten olemassa oleva proseduuri vaatii haettavat aineistot omina parametreinaan, joten metodin saama lista parametriobjekteista täytyy tässä vaiheessa vielä läpikäydä ja asettaa parametriobjektien sisältö proseduurikutsua varten omiksi parametrien arvoiksi.

AddFile- operaation kohdalla käytettävä aineiston sisäänlukua varten oleva tietokantaproseduuri ottaa vastaan lisättävän aineiston Request- sanomassa. Kutsuttaessa tätä proseduuria Web Servicestä Request- sanomalla, kutsuun lisätään SOAP- sanomassa olevan AttachmentPart- liitteen sisältö. Proseduurista palautuu aineiston sisään luvun onnistuessa Responce- sanomassa selkokielinen paluusanoma jossa on koodinumero 0, aineiston koko sekä teksti ok. Aineiston sisään luvun epäonnistuessa Responce- sanomassa on virhekoodi sekä selkokielinen selitys aineiston sisään luvun epäonnistumisen syystä. Tämän Responce- sanoman vastaus luetaan Web Servicessä ja siitä muodostuu addFile- operaation paluuarvo. Tämä paluuarvo viedään Web Serviceä kutsuneen käyttäjän lähettämän Request- sanoman vastauksena lähetettävän Responce- sanoman sisältämään SOAP- sanomaan.

GetFile- operaation kohdalla käytettävä aineiston muodostusta oleva tietokantaproseduuri palauttaa sitä kutsuttavaan Request- pyyntöön Responce-sanoman, jonka sisältönä on haluttu XML- muotoinen aineisto. Tästä Responce-sanoman vastauksena tulevasta aineistosta muodostetaan SOAP AttachmentPart- liite. Liite annetaan getFile- operaation paluuarvona Web Serviceä kutsuneen käyttäjän lähettämän Request- sanoman vastaukseen. Liite tulee siis Web Servicen Responce-sanoman sisältämässä SOAP- sanomassa AttachmentPart- liitteenä.

5 Pohdinta

Projektin toteutus sujui kokonaisuutena hyvin ja opinnäytetyön tilaaja oli tyytyväinen projektin aikana toteutettuun ratkaisuun. Projektin aikataulun suunnittelussa suurimmaksi haasteeksi muodostui se, ettei minulla ollut minkäänlaista kokemusta Web Service- rajapinnan kehityksestä. Tämä johti myös siihen, että projektin eri tehtävien ja niiden toteutukseen vaadittavien asioiden kokonaisuuden hahmottamisessa oli vaikeuksia. Projektin edetessä kokemuksen karttuessa oli helpompi arvioida toteutettavien asioiden kehittämiseen kuluvaa aikaa.

Tärkeänä opetuksena tulevia projekteja varten sisäistin myös projektin vaatimusmäärityksen tarkkuuden korostamisen. Hyvin ja tarkasti suunniteltu projekti on selkeämpi ja nopeampi toteuttaa, sekä projektin eri osa-alueiden toteuttamiseen

vaadittavat resurssit helpompi hahmottaa kuin huonosti määritellyssä projektissa. Tämän projektin osalta en itse osannut määritellä tilaajan kanssa heti projektin alussa projektin toteutuksessa vaadittavia asioita tarpeeksi tarkasti. Projektin toteutus oli määritelty pääpiirteittäin alussa, mutta en itse osannut hahmottaa ja tarkentaa joitain asioita, joiden hahmottaminen heti projektin alkaessa olisi helpottanut omaa työskentelyäni.

Tästä esimerkkinä autentikoinnin ja auktorisoinnin toteutus, jonka toteutus vei projektin teknisestä toteutuksesta yksittäisenä ominaisuutena suurimman ajan. Syynä tähän oli se, etten ollut hahmottanut sitä, etten pystynyt tekemään vaadittavan tarkkaa auktorisointia aineistokohtaisesti suoraan toteuttamassani Login Modulessa. Pelkkä käyttäjätunnuksen ja salasanan tarkistus ei ollut riittävä auktorisointi tarkemmassa projektin edetessä esiin tulleessa vaatimuksessa, vaan auktorisointia varten oli tarkistettava SOAP- sanoman sisällöstä kutsuttu operaatio sekä sille annetut parametrit. Suurin syy mielestäni siihen, etten osannut projektin suunnitteluvaiheessa tilaajan kanssa tarpeeksi tarkasti määritellä vaadittavaa toteutusta, oli kokemattomuus Web Service- rajapinnan kehittämisestä.

Vaikka projektin suunnittelu sekä määrittely eivät toteutuneet niin tarkasti kuin olisi toivottavaa ja projektin toteutukseen kuluvien resurssien hahmottaminen ei perustunut kokemukseen tämänkaltaisen projektin toteutuksesta, onnistui projektin toteutus sovelluksen osalta pienemmässä käytetyssä ajassa kuin siihen oli resursoitu.

Opinnäytetyön kirjalliseen osuuteen resursoitu aika toteutui kutakuinkin juuri niin kuin oli laskettukin. Ainoana ongelmana kirjoittamisen osalta oli minulla se, etten onnistunut käyttämään näitä toteutuneita resursseja projektin alussa määrittelemässäni ajassa. Itse tekninen toteutus oli ajallaan valmis, mutta kirjoittaminen jatkui vielä aluksi suunniteltua projektin päättämistä myöhempään. Tästä syystä projektin päättyminen myöhästyi suunnitellusta ajankohdasta muutamia viikkoja. Suurin syy tähän oli yksinkertaisesti se, etten osannut suunnitella omaa ajankäyttöäni kokonaisuutena niin, että kirjoittamiseen olisi jäänyt tarpeeksi aikaa silloin kun piti.

Kokonaisuutena olin tyytyväinen omaan työskentelyyni toteutuksen kehitystyössä. Projektin hallintaan, vaatimusmäärittelyyn ja ajankäytön suunnitteluun en ollut täysin tyytyväinen ja näihin asioihin aion jatkossa kiinnittää erityistä huomiota tulevissa työprojekteissani.

Jatkokehityksessä Web Serviceä varten toteutetaan erilliset kumppanikäyttäjätunnukset, joita toimeksiantaja, Finnvalli Oy hallinnoi. Finnvalli antaa näille kumppanikäyttäjätunnuksille aika- ja määräkohtaiset rajoitukset aineistojen välitykseen Web Servicestä. Finnvallin ylläpitämän Fivaldi – sovelluksen asiakkaat hallinnoivat tietovirtoja, joita kumppanikäyttäjätunnuksella on oikeus heidän yrityksensä osalta siirtää Web Servicen kautta. Jatkokehityksessä autentikointi ja auktorisointi tapahtuu näitä tallennettuja tietoja vasten. Näille kumppanikäyttäjien salasanoille tullaan toteuttamaan nykyisistä käyttäjien salasanoista poikkeava salaus tietokantaan.

Kun muutokset koskien uusia Web Servicen käyttöön oikeuttavia käyttäjätunnuksia toteutetaan nykyisen sovelluksen palvelunhallintaan, autentikoinnin ja auktorisoinnin tekevässä tietokantaproseduurissa luodaan käyttäjälle keksi, joka oikeuttaa pyydettyjen aineistojen noutoon tai tuomiseen sovelluksen tietokantaan tietyn aikarajan sisällä. Tämä keksi palautetaan tulevaisuudessa messageHandler Java- luokalle koodin lisäksi ja sitä käytetään autentikointi ja auktorisointi tietona kun varsinainen aineiston tuonti tai nouto tehdään. Itse aineistosiirtoon toteutetaan oma proseduuri, jonne käyttäjätunnuksen ja salasanan tilalla lähetetään edellä mainittu autentikoinnin ja auktorisoinnin suorittavan proseduurin luoma keksi.

Tämän projektin osalta ei tehty Web Servicen suorituskyvyn testausta ja jatkokehityksessä on syytä ottaa huomioon mahdolliset suorituskykyongelmat varsinkin suurikokoisten aineistojen siirrossa Web Service- rajapinnan kautta. Jatkokehityksessä suorituskyvyn osalta myös useiden Web Service- rajapintaan samanaikaisesti kohdistuvien pyyntöjen testaus on oltava kattava ennen Web Servicen tuotantoon siirtoa. Tässä projektissa ei toteutettu toistohyökkäyksien kannalta tärkeää WS-Security UsernameToken- elementin sisältävän satunnaismerkkijono tyyppisen nonce-elementin säilyttämistä Web Servicen muistissa ja muistissa olevien saman nonce-

elementin sisältävien kutsujen hylkäystä. Tämä on syytä ottaa huomioon Web Servicen jatkokehityksessä.

Lähteet

Autentikaatio. http://www.ffiec.gov/pdf/authentication_guidance.pdf. Luettu 1.10.2013.

Auktorisointi. http://www.businessdictionary.com/definition/authorization.html. Luettu 1.10.2013.

Callback interface. http://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/Callback.html. Luettu 15.11.2013.

Erl, T.2008. Service-Oriented Architecture. Concepts, Technology and design. PRENTICE HALL. U.S.

HTTP. http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html#sec1. Luettu 2.10.2013.

Java Handler. http://docs.oracle.com/javaee/1.4/api/javax/xml/rpc/handler/Handler.html. Luettu 20.10.2013.

Java merkkijono. http://docs.oracle.com/javase/6/docs/api/java/lang/String.html. Luettu 1.11.2013.

Java Oraclen tietokannassa. http://docs.oracle.com/cd/B28359_01/java.111/b31225/chthree.htm#CACICFFB. Luettu 01.12.2013.

Käyttäjä. http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html#sec1. Luettu 5.11.2013.

OASIS organisaatio. https://www.oasis-open.org/. Luettu 15.10.2013.

Oasis Usernametoken profile. https://www.oasis-open.org/committees/download.php/13392/wss-v1.1-spec-pr-UsernameTokenProfile-01.htm. Luettu 15.10.2013.

Oracle custom login modules. http://docs.oracle.com/cd/B14099_19/web.1012/b14013/loginmod.htm. Luettu 11.11.2013.

Oracle XML Type Operations. http://docs.oracle.com/cd/B12037_01/appdev.101/b10790/xdb04cre.htm. Luettu 01.12.2013.

Papazoglou, M. 2012. Web Service & SOA. Principles and Technology. Second Edition. PEARSON. England.

Request-sanoma. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5. Luettu 5.10.2013.

Responce-sanoma. http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6. Luettu 5.10.2013.

Server. http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html#sec1. Luettu 5.10.2013.

SOAP protokolla. http://www.oracle.com/technetwork/topics/brown-ch17-128273.pdf. Luettu 2.11.2013.

Web Service kuvaus. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#technology. Luettu 5.10.2013.

XML. http://www.w3.org/standards/xml/core. Luettu 2.10.2013.

Liitteet

Liite 1. Web Servicen lähettämä SOAP sanoma virheellisesti muodostetusta SOAP sanomasta.

Liite 2. Web Servicen lähettämä SOAP sanoma autentikoinnin epäonnistumisesta.

Liite 3. Web Servicen lähettämä SOAP sanoma authorisoinnin epäonnistumisesta pyydettyyn aineistotyyppiin.

Liite 4. Client sovelluksen tekemä SOAP sanoma getFile operaatioon.

Liite 5. Web Servicen muodostama SOAP paluusanoma getFile operaation kutsuun (itse sanoma liitteenä samassa Responce sanomassa).

Liite 6. Client sovelluksen tekemä SOAP sanoma addFile operaatioon kutsuun (itse sanoma liitteenä samassa Responce sanomassa).

Liite 7. Web Servicen muodostama SOAP sanoma addFile operaation kutsuun.

Liite 8. Login Modulen määrittely Web Serviceen Oracle Enterprice Managerilla.

Liite1. Web Servicen lähettämä SOAP sanoma virheellisesti muodostetusta SOAP sanomasta.

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
   <env:Header/>
   <env:Body>
      <env:Fault>
         <faultcode xmlns="">env:Client</faultcode>
         <faultstring xmlns="">Element: schema not val-
id!</faultstring>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

Liite 2. Web Servicen lähettämä SOAP sanoma autentikoinnin epäonnistumisesta.

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
   <env:Header/>
   <env:Body>
      <env:Fault xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <faultcode xmlns="">wsse:FailedAuthentication</faultcode>
         <faultstring xmlns="">The security token could not be au-
thenticated or authorized</faultstring>
         <faultactor xmlns=""/>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

Liite 3. Web Servicen lähettämä SOAP sanoma authorisoinnin epäonnistumisesta pyydettyyn aineistotyyppiin.

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
   <env:Header/>
   <env:Body>
      <env:Fault xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <faultcode xmlns="">wsse:FailedAuthentication</faultcode>
         <faultstring xmlns="">The security token could not be au-
thenticated or authorized, Authorisoinnin epäonnistumisen syy:
Virhekoodi 40  Ei oikeutta aineistoon
(yritysrekisteri)</faultstring>
         <faultactor xmlns=""/>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

Liite 4. Client sovelluksen tekemä SOAP sanoma getFile operaatioon.

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
   <env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
env:mustUnderstand="1">
         <wsse:UsernameToken xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
            <wsse:Username>fvdemo-webservice</wsse:Username>
            <wsse:Password Type="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">jKm+c9oewFHfTSml86q+VMkl1vQ=</wsse:Password>
            <wsse:Nonce>KueywnQzn1i9iUT8xLonNw==</wsse:Nonce>
            <wsu:Created Valu-
eType="http://www.w3.org/2001/XMLSchema/dateTime">2013-12-
30T07:48:28Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
   </env:Header>
   <env:Body>
      <ns0:getFile>
         <schema
            xmlns="">valli</schema>
         <businessId xmlns="">300020</businessId>
         <ParameterObject xsi:type="ns1:arrayList" xmlns="">
            <ns1:item xsi:type="ns0:parameterObject">
               <ns0:parameter>p_yrek</ns0:parameter>
            </ns1:item>
         </ParameterObject>
      </ns0:getFile>
   </env:Body>
</env:Envelope>
```

Liite 5. Web Servicen muodostama SOAP paluusanoma getFile operaation kutsuun (itse sanoma liitteenä samassa Responce sanomassa).

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd">
          <env:Header>
                    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
env:mustUnderstand="1"/>
          </env:Header>
          <env:Body>
                    <ns0:getFileResponse>
                              <result
xmlns="">cid:ID1@result</result>
                    </ns0:getFileResponse>
          </env:Body>
</env:Envelope>
```

Liite 6. Client sovelluksen tekemä SOAP sanoma addFile operaatioon kutsuun (itse sanoma liitteenä samassa Responce sanomassa).

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
          <env:Header>
                    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
env:mustUnderstand="1">
                              <wsse:UsernameToken
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
                                        <wsse:Username>fvdemo-
webservice</wsse:Username>
                                        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">eL4b3T8cTy9vJffOtjwEgkGVx70=</wsse:Password>
          <wsse:Nonce>PH8REpsSYPn/cXCJe3td2Q==</wsse:Nonce>
                                        <wsu:Created Valu-
eType="http://www.w3.org/2001/XMLSchema/dateTime">2013-12-
30T08:23:08Z</wsu:Created>
                              </wsse:UsernameToken>
                    </wsse:Security>
          </env:Header>
          <env:Body>
                    <ns0:AddFile>
                              <schema xmlns="">valli</schema>
                              <businessId
xmlns="">300030</businessId>
                              <ParameterObject
xsi:type="ns1:arrayList" xmlns="">
                                        <ns1:item
xsi:type="ns0:parameterObject">
          <ns0:parameter>p_tuotteet</ns0:parameter>
                                        </ns1:item>
                              </ParameterObject>
                              <incomingMessage
xmlns="">cid:ID1@incomingMessage</incomingMessage>
                    </ns0:AddFile>
          </env:Body>
</env:Envelope>
```

Liite 7. Web Servicen muodostama SOAP paluusanoma addFile operaation kutsuun.

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://finnvalliwebservicesendpoint/"
xmlns:ns1="http://www.oracle.com/webservices/internal/literal"
xmlns:ns2="http://ws-i.org/profiles/basic/1.1/xsd">
   <env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
env:mustUnderstand="1"/>
   </env:Header>
   <env:Body>
      <ns0:AddFileResponse>
         <result xmlns="">0 2519 ok</result>
      </ns0:AddFileResponse>
   </env:Body>
</env:Envelope>
```

Liite 8. Login Modulen määrittely Web Serviceen Oracle Enterprice Managerilla.