Juha Koponen

Web Services pankkiyhteyspalvelussa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

29.04.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Juha Koponen Web Services pankkiyhteyspalvelussa 40 sivua + 1 liite 29.4.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoliikennetekniikka
Ohjaaja(t)	Tuotepäälliiko Olli Johansson Osaamisaluepäällikkö Janne Salonen

Tässä opinnäytetyössä perehdytään Web Service -kanavan toteutukseen palveluntarjoajan näkökulmasta. Tavoitteina oli asiantuntemuksen laajentaminen palvelua ylläpitävässä organisaatiossa, dokumentaation päivittäminen ja palveluprosessien kehittäminen.

Työssä perehdytään seikkaperäisesti xml-dokumenttien prosessointiin sekä Web Services -standardeihin. Opinnäytetyössä analysoidaan Web Services -pankkiyhteysohjelmia tuottavien ohjelmistotalojen palvelupyyntöjä ja pyritään tunnistamaan yleisimmin ongelmia aiheuttaneita kohtia, joihin voidaan puuttua dokumentaatiota ja sisäisiä prosesseja kehittämällä.

Työn toteutuksen pohjalta saatiin selkeytettyä palveluprosessia ja parannettua ohjelmistotaloille tarjottavia testausmahdollisuuksia. Työn tuloksien pohjalta laadittua koulutusmateriaalia tullaan hyödyntämään organisaation sisäisessä koulutuksessa.

Avainsanat	Web services, XML, SOAP,WSDL, PKI

Author(s) Title Number of Pages Date	Juha Koponen Web Services application for financial usage 40 pages + 1 appendix 29 April 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Telecommunication
Instructor(s)	Olli Johansson, Product Manager Janne Salonen, Principal Lecturer

Purpose of this Bachelor's thesis was to study the implementation of the Web Service channel in service provider's point of view. The aim was to get wider service expertise in organization, updating documentation and streamline service processes.

In the thesis XML document processing and Web Services standards was researched and service request from software companies were analyzed to identify the most common problems of the points of which could be addressed in the documentation and internal processes.

As a result of the thesis several internal processes got simplified and testing opportunities offered for software companies got enhanced. Internal material produced during the thesis will be used for training inside the organization.

Keywords	Web services, XML, WSDL, SOAP, PKI

Sisällys

1 Johdanto

Sähköistä pankkiasiointia on pitkään hoidettu pankkikohtaisilla tai kansallisesti kehitetyin yhteystekniikoin, kuten PATU-kanava. Nämä pankkien kehittämät yhteystavat alun perin pohjautuivat soittosarjoihin piirikytkentäisen verkon yli. Myöhemmin on siirrytty liikennöimään IP-verkon ftp-protokollaa käyttäen, mutta joitain alkuperäisen tekniikan asettamia rajoituksia on kulkeutunut mukana tähän päivään saakka. [1.]

Web Services valittiin seuraavan sukupolven tekniikaksi yritysasiakkaiden ja pankin väliseen kommunikointiin. Uusi tekniikka pohjautuu täysin kansainvälisiin standardeihin. Tämä helpottaa niin pankkien omaa sovelluskehitystä kuin asiakasohjelmistojen toimittajia esimerkiksi integroimaan pankkiyhteyspalvelut ERP- ja reskontraohjelmistoihinsa.[2.]

Web Services perustuu XML- ja http-standardeihin. XML mahdollistaa monimutkaisten viestien ja komentojen välittämisen erilaisten ympäristöjen ja eri ohjelmointikielillä toteutettujen ohjelmistojen välillä. Http puolestaan on käytetyin internetprotokolla. Pankkiyhteysliikenteessä käytetään aina SSL/TSL-salattua http-yhteyttä.

Web Servicen peruselementtejä ovat:

- SOAP (Simple Object Acces Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description, Discovery and Integration).

Web Services -kanavaan siirtyminen on yksi osa pankkien SEPA eli yhtenäisen euromaksualueen migraatiota. [2; 3.]

Opinnäytetyön toimeksiantaja Oy Samlink Ab on haastavien tietojärjestelmien ratkaisu- ja palvelutoimittaja. Konsernin liikevaihto oli vuonna 2013 noin 93,8 miljoonaa euroa ja henkilöstön määrä ylitti 500 henkilön rajan. Opinnäytetyötä tehtiin vakituisessa työsuhteessa ja tavoitteena oli kehittää organisaation sisäisiä prosesseja, dokumentaatiota ja laajentaa osaamista asiakaspalvelussa toimivien toimihenkilöiden keskuudessa.

2 XML

2.1 Mitä on XML?

XML eli Extensible Markup Language on hyvin samankaltainen www-sivujen luontiin tarkoitetun HTML-kielen kanssa, mutta eri käyttötarkoitukseen. HTML on kehitetty näyttämään tietoa keskittyen siihen, miltä esitetty tieto näyttää ja XML on puolestaan tarkoitettu tiedon välittämiseen ja tallentamiseen, jossa pääpainona on, mitä välitettävä tai tallennettava tieto on. XML-kielessä ei ole ennalta määrätyn nimisiä elementtejä, vaan elementit voi nimetä kuvaamaan sen sisältämää tietoa. [4; 5.]

2.2 XML-tiedoston rakenne

XML-tiedosto rakentuu tiedoston prosessointikäskyistä, juurielementistä, sen lapsielementeistä ja attribuuteista. Juurielementin attribuutteina usein määritellään dokumentissa käytettävät nimiavaruudet sekä dokumentin rakennetta ja elementtien tietotyyppejä ohjaava mallitiedosto eli xml-schema. Alaluvuissa näihin tutustutaan tarkemmin yksinkertaisten esimerkkien avulla.

2.2.1 XML-dokumentti

Kuviossa 1 on esitetty yksinkertaisen XML-dokumentin rakenne.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Videovuokraamo>
  - <Elokuva Id="3526">
      <Nimi>Kauhea Kankkunen III</Nimi>
      <Alkuperäinen_nimi>The Hangover Part III</Alkuperäinen_nimi>
      <Julkaisuvuosi>2013</Julkaisuvuosi>
      <Ohjaus>Todd Phillips</Ohjaus>
      <Pääosissa>Bradley Cooper, Ed Helms, Zach Galifianakis, Ken Jeong, Heather Graham</Pääosissa>
      <Kesto yksikkö="min">100</Kesto>
      <Genre>Komedia</Genre>
      <Ikäraja>K-12</Ikäraja>
    </Elokuva>
  - <Elokuva Id="3527">
      <Nimi>8-pallo</Nimi>
      <Alkuperäinen_nimi />
      <Julkaisuvuosi>2013</Julkaisuvuosi>
      <Ohjaus>Aku Louhimies</Ohjaus>
      <Pääosissa>Jessica Grabowsky, Eero Aho, Pirkka-Pekka Petelius, Mikko Leppilampi</Pääosissa>
      <Kesto yksikkö="min">103</Kesto>
      <Genre>Draama</Genre>
      <Ikäraja>K-16</Ikäraja>
    </Elokuva>
</Videovuokraamo>
```

Kuvio 1. XML-dokumentin rakenne

Ensimmäisellä rivillä on prosessointikäsky, jossa määritellään dokumentin yleisiä ominaisuuksia, kuten XML-versio ja käytetty merkistökoodaus. Prosessointikäskyillä voidaan myös määritellä esimerkiksi xml-tyylitiedosto, jos dokumenttia on tarkoitus tarkastella selaimella tai tulostaa paperille. Toisella rivillä on dokumentin juurielementti Videovuokraamo. XML-dokumentissa voi olla vain yksi juurielementti, jonka sisälle koko dokumentti rakentuu. Juurielementin alla esimerkissä löytyy Elokuvaelementti, jonka alla on kahdeksan elokuvaa kuvaavaa lapsielementtiä.

Elokuvaelementillä on attribuutti Id, jolla tässä on kuvattu elokuvan arkistointinumeroa. XML-attribuuteilla voi elementtien ohella kuvata tietoa. XML-attribuuttien heikkous on se, että ne eivät voi sisältää useita arvoja toisin kuin elementit, joille voi lisätä lapsielementtejä. XML-attribuuteilla on kuitenkin hyvä kuvata tietoa, joka ei olennaisesti liity elementtiin tai ilmaista ,minkä tyyppistä tietoa elementti sisältää. Elokuvan lapsielementissä, kesto on attribuuttina yksikkö, jolle on annettu kiinteä arvo min. Tällä viitataan, että elokuvan kesto on ilmoitettu minuutteina. [5.]

2.2.2 XML Namespaces

Koska XML-kielessä elementtien nimet ovat ohjelmistokehittäjän määrittelemät, voi nimien välillä usein syntyä ristiriita eri ohjelmistoissa ja sovelluksissa.

Kuviossa kaksi XML dokumentti sisältää HTML taulukon.

```
- <table>
  - <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
  </table>
```

Kuvio 2. XML sisältää HTML-taulukon.

Kuviossa kolme on XML-dokumentti, joka sisältää tietoa pöydästä ja sen ominaisuuksista.

```
- <table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
  </table>
```

Kuvio 3. XML kuvaa pöydän ominaisuuksia.

Jos nämä edellä kuvatut XML-dokumentit yhdistetään, tulee konflikti kahden elementin välille. Molempien elementtien nimi on table, mutta niiden merkitys ja sisältö ovat täysin erilaiset. Tällaista XML-dokumenttia käsittelevä sovellus ei voi tietää, kuinka käsitellä näitä elementtejä.

Ratkaisu elementtien välisiin nimeämisongelmiin on käyttää XML Namepace:ja eli nimiavaruuksia. Elementin nimeen lisätään viittaus, mihin nimiavaruuteen elementti kuuluu. Nimiavaruus täytyy aina määritellä xmlns-attribuutilla elementille. Määrittely yleensä tehdään jo juurielementissä koko dokumentille. Kuviossa neljä on juurielementissä määritelty kaksi nimiavaruutta, joita on hyödynnetty erottamaan kaksi <table>-nimistä elementtiä toisistaan.

```
- <root xmlns:h="http://www.w3.org/TR/html4/" xmlns:f="http://www.w3schools.com/furniture">
  - <h:table>
    - <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
      </h:tr>
    </h:table>
  - <f:table>
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
    </f:table>
  </root>
```

Kuvio 4. Esimerkki nimiavaruuksien käytöstä

Nyt dokumenttia käsittelevällä ohjelmalla ei ole ongelmia erottaa <table>-elementtejä, koska ne ovat eri nimiavaruuksissa. Vaikka nimialueen määrittelevä atribuutti sisältää aina jonkin URI:n, ei tästä osoitteesta kuitenkaan tarkasteta mitään. Tarkoitus on vain antaa nimiavaruudelle täysin uniikki nimi. Hyvin yleisenä tapana on viitata URI:ssa internetsivulle, joka sisältää tietoa nimiavaruudesta. [4; 5.]

2.2.3 XML Schema

XML-mallitiedostossa eli XML-schemassa määritellään XML-dokumentin rakenne. Mallitiedostoa hyödynnetään vastaanotetun XML-dokumentin sisällön ja rakenteen tarkastuksessa. Tarkastamalla rakenne voidaan varmistua, että dokumentti sisältää järjestelmän mukaista sisältöä. Käytännön esimerkkinä SEPA-maksuaineistosta tarkastetaan, että kaikki maksun suorittamiseen vaaditut tiedot on lähetetyssä aineistossa.

XML Schemalla voidaan määritellä:

- mitä elementtejä voi esiintyä dokumentissa
- mitä attribuutteja voi esiintyä dokumentissa
- mitkä elementit ovat lapsielementtejä
- lapsielementtien määrän ja järjestyksen
- voiko elementti olla tyhjä tai sisältää tekstiä
- elementtien ja attribuuttien tietotyypit
- elementtien ja attribuuttien oletus- ja kiinteät arvot.

Elementtien tietotyypit ovat ehkä XML-mallitiedostojen käytön suurimpia hyötyjä. Kun siirretään tietoa kahden järjestelmän välillä, voi syntyä ristiriitaisia tulkintoja sisällöstä. Oletetaan, että dokumentissa vaaditaan tieto päivämäärästä elementissä <pvm>. Vastaanotetussa dokumentissa elementin sisältä on 05-09-2013. Tämä tulkitaan suomalaisen merkintätavan mukaan syyskuun viidenneksi päiväksi tai amerikkalaisella merkintätavalla yhdeksänneksi toukokuuta. Kun elementille määritellään tietotyyppi date <pvm type=”date”>2013-09-05<pvm>, voidaan varmistua, että vastaanottaja tulkitsee lähettäjän antaman päivämäärän niin kuin lähettäjä on sen tarkoittanut, koska XML-tietotyyppi ”date” vaatii päivämäärän esitettävän muodossa VVVV-KK-PP.

Lisätään kuvion yksi videovuokraamo esimerkkiin nimiavaruus xsi ja määritellään mallitiedoston eli scheman sijainti.

```
<?xml version="1.0" encoding="utf-8" ?>
- <xsi:Videovuokraamo xmlns:xsi="http://www.vv.fi/XMLSchema-instance" xsi:schemaLocation="http://www.vv.fi
    vvschema.xsd">
  - <xsi:Elokuva Id="3526">
      <xsi:Nimi>Kauhea Kankkunen III</xsi:Nimi>
      <xsi:Alkuperäinen_nimi>The Hangover Part III</xsi:Alkuperäinen_nimi>
      <xsi:Julkaisuvuosi>2013</xsi:Julkaisuvuosi>
      <xsi:Ohjaus>Todd Phillips</xsi:Ohjaus>
      <xsi:Pääosissa>Bradley Cooper, Ed Helms, Zach Galifianakis, Ken Jeong, Heather Graham</xsi:Pääosissa>
      <xsi:Kesto yksikkö="min">100</xsi:Kesto>
      <xsi:Genre>Komedia</xsi:Genre>
      <xsi:Ikäraja>K-12</xsi:Ikäraja>
    </xsi:Elokuva>
  - <xsi:Elokuva Id="3527">
      <xsi:Nimi>8-pallo</xsi:Nimi>
      <xsi:Alkuperäinen_nimi />
      <xsi:Julkaisuvuosi>2013</xsi:Julkaisuvuosi>
      <xsi:Ohjaus>Aku Louhimies</xsi:Ohjaus>
      <xsi:Pääosissa>Jessica Grabowsky, Eero Aho, Pirkka-Pekka Petelius, Mikko Leppilampi</xsi:Pääosissa>
      <xsi:Kesto yksikkö="min">103</xsi:Kesto>
      <xsi:Genre>Draama</xsi:Genre>
      <xsi:Ikäraja>K-16</xsi:Ikäraja>
    </xsi:Elokuva>
  </xsi:Videovuokraamo>
```

Kuvio 5. Videovuokraamoesimerkkiin lisätty nimiavaruus ja scheman määrittelyt

Kuten kuviosta viisi voi huomata, mallitiedoston tiedostoon viitataan juurielementissä schemaLocation-attribuutilla. Tällä atribuutilla on kaksi arvoa, jotka on erotettu välilyönnillä. Ensimmäinen arvo http://www.vv.fi kertoo käytettävän nimiavaruuden ja toinen arvo vvschema.xsd kertoo mallitiedoston, jota kyseisessä nimiavaruudessa sovelletaan.

```
xsi:schemaLocation="http://www.vv.fi vvschema.xsd">
```

Kuvio 6. schemaLocation-attribuutti tarkemmin

Mallitiedoston juurielementtinä on aina <schema>. Schemassa määritetyille elementeille määritellään aina elementin tyyppi sekä mahdolliset attribuutit ja rajoitukset. Käytettävät elementtityypit ovat yksinkertainen tyyppi simpleType tai kompleksinen tyyppi complexType. SimpleType-elementti voi sisältää vain tekstiä tai numeroita. Se ei voi sisältää attribuutteja tai muita elementtejä. Jos elementillä on attribuutteja, sen tyyppi on aina complexType, mutta itse attribuutit määritellään aina yksinkertaisiksi. Rajoituksilla, restrictions, voidaan määrittää elementin sisällölle rajoituksia. Kuviossa seitsemän on videovuokraamodokumentin mallitiedosto vvschema.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
- <xs:schema xmlns:xsi="http://www.vv.fi/XMLSchema-instance" targetNamespace="http://www.vv.fi/XMLSchema-instance"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:element name="Videovuokraamo">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="xsi:Elokuva" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="schemaLocation" form="qualified" use="required"/>
      </xs:complexType>
    </xs:element>
  - <xs:element name="Elokuva">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="xsi:Nimi"/>
          <xs:element ref="xsi:Alkuperainen_nimi"/>
          <xs:element ref="xsi:Julkaisuvuosi"/>
          <xs:element ref="xsi:Ohjaus"/>
          <xs:element ref="xsi:Paaosissa"/>
          <xs:element ref="xsi:Kesto"/>
          <xs:element ref="xsi:Genre"/>
          <xs:element ref="xsi:Ikaraja"/>
        </xs:sequence>
        <xs:attribute name="Id" use="required" type="xs:integer"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="Nimi" type="xs:string"/>
    <xs:element name="Alkuperainen_nimi" type="xs:string"/>
    <xs:element name="Julkaisuvuosi" type="xs:integer"/>
    <xs:element name="Ohjaus" type="xs:string"/>
    <xs:element name="Paaosissa" type="xs:string"/>
  - <xs:element name="Kesto">
    - <xs:complexType>
      - <xs:simpleContent>
        - <xs:extension base="xs:integer">
            <xs:attribute name="yksikko" use="required" type="xs:string" fixed="min"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="Genre" type="xs:string"/>
  - <xs:element name="Ikaraja">
    - <xs:simpleType>
      - <xs:restriction base="xs:string">
          <xs:enumeration value="Sallittu"/>
          <xs:enumeration value="K-7"/>
          <xs:enumeration value="K-12"/>
          <xs:enumeration value="K-16"/>
          <xs:enumeration value="K-18"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
</xs:schema>
```

Kuvio 7. Videovuokraamodokumentin mallitiedosto

Mallitiedostossa ensimmäisenä on määritelty dokumentin juurielementti Videovuokraamo. Videovuokraamoelementti on tyypiltään kompleksinen, sillä se sisältää lapsielementin Elokuva sekä attribuutin schemaLocation, jolla xml-dokumentissa viitataan mallitiedostoon. Seuraavana skeemassa määritellään Elokuvaelementti ja sen rakenne. Elementtillä on kahdeksan lapsielementtiä sekä yksi attribuutti ja on näin ollen myös tyypiltään kompleksinen elementti. Lapsielementteihin on viitattu <xs:element ref="xsi:Nimi"/>-rakenteella. Viittaus ei määrittele esimerkin xsi:Nimi-elementtiä, ilmaisulla vain viitataan toisaalla määriteltyyn elementtiin. Kaikki kahdeksan lapsielementtiä on kerätty sequence –tag:ien sisään, jolloin niitä käsitellään kokonaisuutena eli kaikkien lapsielementtien tulee esiintyä tarkastettavassa xml-dokumentissa ja mallitiedoston mukaisessa järjestyksessä. Elokuva elementin attribuutti Id, on määritelty integer-tyyppiseksi eli attribuutin arvona voi olla vain kokonaislukuja. Attribuutti on määritelty pakolliseksi kaikille elokuvaelementeille. Elokuvan tiedot sisältävät kahdeksan elementtiä ovat Kesto-elementtiä lukuun ottamatta tyypiltään yksinkertaisia elementtejä,

sillä ne sisältävät vain tekstiä tai numeroita. Kesto-elementillä on attribuuttiyksikkö, joka tekee elementistä kompleksisen, vaikka sen sisältönä onkin vain kokonaisluku. Attribuutti-yksikkö sisältää tekstiä ja sille on annettu kiinteä arvo: min. Ikärajaelementin sisällölle on asetettu rajoituksia. Restriction tag:ien sisään on lueteltu sallituiksi arvoiksi suomessa Mediakasvatus- ja kuvaohjelmakeskuksen määrittelemät elokuvien ikärajat string, eli merkkijono tyyppisinä arvoina. [4; 5; 6.]

3 Web Services

Web Service on ohjelmistojärjestelmä, jonka avulla keskenään yhteensopivat tietokoneet voivat kommunikoida tietoverkon yli. Yksinkertaistettuna Web Service tarkoittaa WWW-pohjaisia ohjelmointirajapintoja. Web Servicessä palvelun tarjoaja ja käyttäjä kommunikoivat keskenään erilaisten XML-pohjaisten protokollien avulla. Protokolliin kuuluvat aiemmin mainitut kolme komponenttia: SOAP Simple Object Access Protocol, WSDL Web Services Description Language sekä UDDI Universal Description Discovery and Integration. Tässä työssä perehdytään vain WSDL- ja SOAP-standardeihin, sillä pankkiyhteyskäyttöön tarkoitettua Web Services -palvelua ei ole julkaistu hajautetun UDDI-rekisterin kautta, vaan pankit ovat päätyneet jakamaan kehittäjille tarkoitetun wsdl-dokumentin ja muun tarpeellisen dokumentaation esimerkiksi omien nettisivujensa välityksellä. UDDI:a voisikin yksinkertaisimmillaan kuvata kehittäjille tarjottavaksi rekisteriksi, jota palveluntarjoaja voi käyttää palvelunsa julkaisemiseen.

Esimerkkinä verkossa julkaistavasta palvelusta voisi olla valuuttamuunnin. Kehittäjä, joka on luonut valuuttamuunninohjelmiston, haluaa tarjota palvelua käyttäjille verkon yli. Palvelun tarjoajan pitää pystyä kuvaamaan käyttäjälle esimerkiksi, mitä toimintoja palvelussa on käytössä ja kuinka niitä kutsutaan, mitä protokollia tiedonvälitykseen käytetään, missä muodossa ohjelmistolle lähetettävän datan tulee olla, minkä muotoisen vastauksen ohjelmisto antaa käyttäjälle ja niin edelleen. Jotta mahdolliset valuuttamuuntimen käyttäjät tietäisivät palvelun olemassa olosta ja kuinka sitä käytetään, tulee kehittäjän vielä pystyä välittämään kaikki tarvittavat tiedot käyttäjän saataville. [2; 7.]

3.1 WSDL Web Services Description Language

Web Services Description Language on xml-mekintäkieleen pohjautuva määrittelykieli, joka kehitettiin kuvaamaan ja julkaisemaan web-palvelun toiminnot, sanomaformaatit sekä käytettävät protokollat, standardoidulla tavalla. WSDL:n on alun perin kehittänyt Microsoft, Ariba ja IBM. WSDL:n versio 1.1:n määrittelyt esitettiin W3C:lle, joka hyväksyi ehdotuksena (”Note”) ja julkaisi sen sivuillaan maaliskuussa 2001. Kaksikymmentäkaksi muuta yhtiötä liittyi alkuperäisten kehittäjien ehdotukseen. Vaikka WSDL:n määrittely on edelleen vain julkaistu esitys, eikä W3C ole sitoutunut mitenkään dokumentin sisältöön, voidaan WSDL:ää sen laajan tuen ja esityksen lähes yksimielisyyden vuoksi pitää käytännössä virallisena tapana määritellä Web Services -palvelu [7; 8] Taulukossa 1 on W3C:n teknisten raporttien tasot.

Taulukko 1. W3C:n teknisten raporttien tasot [9.]

Nimi	**Lyhenne**	**Kuvaus**
Note	-	Päivätty, jukinen dokumentti, johon W3C ei ole kuitenkaan sitoutunut tai joka ympärille ei ole vielä perustettu työryhmää. Ei virallinen taso
Working Draft	WD	Ensimmäinen virallinen taso teknisille dokumneteille. Dokumentissa kuvatun aiheen ympärille on muodostettu työryhmä, ja dokumnetti on voimakkaassa kehitysvaiheessa.
Last Call Working Draft	-	Dokumentin sisällöstä on päästy yhteisymmärrykseen työryhmän sisällä ja kommenteille on asetettu aikaraja
Candidate Recommendation	CR	Tässä vaiheessa dokumnettiin ei tule enää suuria muutoksia, mutta dokumentin sisältöä testataan työryhmissä käytännön toteutuksin.
Proposed Recommendation	PR	Dokumentissa kuvatusta standardiehdotuksesta on olemassa vähintäänyksi, mielellään kuitenkin vähintään kaksi toimivaa käytännön toteutusta.
Recommendation	REC	Standardiehdotelmasta on tullut W3C:n virallinen suositus. Jos dokumentti pääsee tälle tasolle, se on hyväksytty ja siitä voidaan katsoa olevan laajaa käytännön hyötyä

WSDL:ssä web servicessä välittettävien sanomien rakenne kuvataan yleensä yhdellä tai useammalla xsd-tyyppimäärittelyllä. WSDL-elementeillä kuvataan operaatiot, mitä vastaanotetulle sanomalle tulisi suorittaa, jotta datan vastaanottaja osaa prosessoida sen käyttäjän toivomalla tavalla. Web Servicesin määrittely sisältää myös käytettävän protokollan ja osoitteen, jotta käyttäjä tietää, kuinka sanoma tulee lähettää web-palvelulle.[7.]

Taulukko 2. Taulukossa on kuvattu wsdl:n tärkeimmät elementit kootusti.

Termi	Elementti	Selite
Datatyypit	types	Yleensä XSD-tyyppimäärityksiin perustuva viestinvälityksessä käytetyn tietotyypin ilmaisu.
Viesti	message	Abstraktia, tietotyypeiltään määrättyä sovellusten välillä kulkevaa tietoa.
Operaatio	operation	Asbstrakti kuvaus operaatiosta tietylle viestille.
Porttityyppi	portType	Abstrakti määrittely operaatioista, jotka liittyy tiettyyn päätepisteeseen; määrittelee joukon operaatioita sidosta varten.
Sidos	binding	Sitoo porttityypillä määritellyt operaatiot ja viestit todelliseen protokollaan ja sanomatyyppiin.
Portti	port	Liittää sidoksen palvelun verkko-osoitteeseen.
Palvelu	service	Määritteleen palveluun kuuluvat portit ja mahdolliset määrittellyyn liittyvät laajennukset.

Edellisen taulukon termit muuttuvat konkreettisemmiksi, kun tarkastellaan varmennepalvelun WSDL-dokumenttia (Liite 1). Koska dokumentit ovat yleensä melko pitkiä ja tottumattomille silmille rakennetta on hankala hahmottaa, voidaan varmennepalvelun wsdl-dokumenttia tutkia osakokonaisuus kerrallaan.

WSDL:n elementit muodostavat viisi selkeää osakokonaisuutta, joihin dokumentin voi jakaa.

- **types**: Tässä osassa määritellään ne tietotyypit, joita käytetään myöhemmin määrittelemään viestin rakennetta.
- **message**: Message-lohkoissa määritellään viestit, joita voidaan välittää sovellusten välillä kyseisen palvelun kautta. Viestit ovat rakenteellisia ja käyttävät joko lohkon sisällä tai types -osassa määritettyjä tietotyyppejä.
- **portType**: Tässä osassa määritellään palvelun tukemat operaatiot. Jokainen operaatio koostuu sekä syöte- että tulosteviestistä (input message, output message).
- **binding**: Sidososassa määritellään tekniset yksityiskohdat viestien välitykseen. Sidoksella määritellään esimerkiksi viestikohtaisesti, mitä esitystapaa, data format, käytetään. WSDL versio 1.1 määrityksessä sidoksia SOAP:lle, HTTP:lle ja MIME:lle
- **service**: Service-lohko sisältää porttimäärityksiä, jotka kertovat, mistä verkko-osoitteesta kuvatut palvelut löytyvät. [7; 9.]

3.1.1 Tietotyypit – type -elementti

WSDL-dokumenteissa hyödynnetään runsaasti XML:n nimiavaruuksia. WSDL-versio 1.1 on nimetty kahdeksan vakio nimiavaruutta ja nimiavaruus tns, this nameSpace, jolla viitataan kyseiseen dokumenttiin. Varmennepalvelun wsdl:ssä näistä nimiavaruuksista on käytössä viisi: xsd, soap, soapenv, wsdl ja tns. Kuviossa 8 on wsdl-dokumentin juurielementti definitions, dokumentissa käytetyt nimiavaruudet ja osa varmennepalvelun datatyypeistä.

```
<?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://mlp.op.fi/OPCertificateService" targetNamespace="http://mlp.op.fi/OPCertificateService">
    - <wsdl:types>
        - <xsd:schema targetNamespace="http://mlp.op.fi/OPCertificateService" attributeFormDefault="qualified"
          elementFormDefault="qualified">
            - <xsd:complexType name="CertificateRequestHeader">
                - <xsd:sequence>
                    <xsd:element name="SenderId" nillable="false" type="xsd:string"/>
                    <xsd:element name="RequestId" nillable="false" type="xsd:string"/>
                    <xsd:element name="Timestamp" nillable="false" type="xsd:dateTime"/>
                  </xsd:sequence>
              </xsd:complexType>
            - <xsd:complexType name="CertificateResponseHeader">
                - <xsd:sequence>
                    <xsd:element name="SenderId" nillable="false" type="xsd:string"/>
                    <xsd:element name="RequestId" nillable="false" type="xsd:string"/>
                    <xsd:element name="Timestamp" nillable="false" type="xsd:dateTime"/>
                    <xsd:element name="ResponseCode" nillable="true" type="xsd:string"/>
                    <xsd:element name="ResponseText" nillable="true" type="xsd:string"/>
                  </xsd:sequence>
              </xsd:complexType>
            - <xsd:complexType name="GetCertificateRequest">
                - <xsd:sequence>
                    <xsd:element name="RequestHeader" nillable="false" type="tns:CertificateRequestHeader"/>
                    <xsd:element name="ApplicationRequest" nillable="false" type="xsd:base64Binary"/>
                  </xsd:sequence>
              </xsd:complexType>
            - <xsd:complexType name="GetCertificateResponse">
                - <xsd:sequence>
                    <xsd:element name="ResponseHeader" nillable="false" type="tns:CertificateResponseHeader"/>
                    <xsd:element name="ApplicationResponse" nillable="false" type="xsd:base64Binary"/>
                  </xsd:sequence>
              </xsd:complexType>
            - <xsd:complexType name="CertificateServiceFaultDetail">
                - <xsd:sequence>
                    <xsd:element name="category" type="xsd:string" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="code" type="xsd:string" maxOccurs="1" minOccurs="0"/>
                  </xsd:sequence>
              </xsd:complexType>
          </xsd:schema>
        - <xsd:schema targetNamespace="http://mlp.op.fi/OPCertificateService" attributeFormDefault="qualified"
          elementFormDefault="qualified">
              <xsd:element name="getCertificatein" type="tns:GetCertificateRequest"/>
              <xsd:element name="getCertificateout" type="tns:GetCertificateResponse"/>
              <xsd:element name="certificateServiceFaultElement" type="tns:CertificateServiceFaultDetail"/>
          </xsd:schema>
    </wsdl:types>
```

Kuvio 8. Esimerkki varmennepalvelun tietotyypeistä

Types-elementissä määritellään sanomilla käytettävät tietotyypit. Yleensä tietotyypit määritellään käyttäen XML-Schema, kuten varmennepalvelunkin tapauksessa. WSDL ei ole sidottu XML-Schemaan ja tietotyyppien määrittely voidaan tehdä muillakin tekniikoilla, mutta tässä työssä paneudutaan vain xml-schemaan pohjautuvaan ratkaisuun.

Varmennepalvelun CertificateRequest- ja –ResponseHeader-tyyppeihin on varattu elementit lähettäjän tietoja sekä aikaleimaa varten. Response-sanomalle on varattu elementit myös vastauskoodia ja vastaustekstiä varten. Seuraavat kaksi tietotyyppiä GetCertificateRequest ja –Response sisältävät Request- tai ResponseHeaderin, joiden sisältä edellä määriteltiin, lisäksi ApplicationRequest tai –Response elementin varsinaista hyötykuormaa varten. ApplicationRequest tai –Responce elementin tietotyyppinä on xsd:base64Binary, eli elementin sisältö on RFC 2045 -standardin mukaisesti Base64-koodattua dataa. CertificateServiceFault-tietotyyppi sisältää elementit järjestelmävirhesanomille.

Jälkimmäisessä schema lohkossa on liitetty ylempänä määritetyt GetCertificateRequest-, -Response ja CertificateServiceFault-tietotyypit omiin isäelementteihinsä GetCertificateRequest- tyyppi getCertificatein-elementtiin ja niin edelleen. [7; 9.]

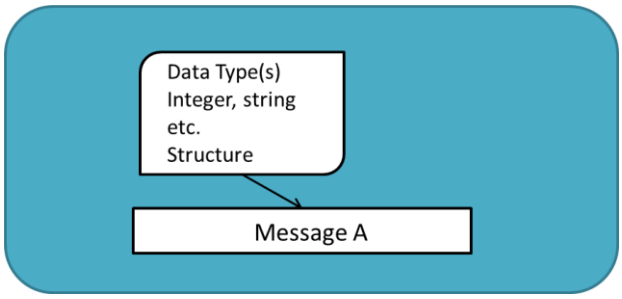
3.1.2 Viestit – message-elementti

WSDL-dokumentissa voi olla useita message-elementtejä, joista jokainen määrittelee yhden viestin, joita sovellusten välillä voidaan välittää. Message-elementit ovat melko yksinkertaisia. Kuviossa 9 on kuvattu osa varmennepalvelun viesteistä, message.

```
- <wsdl:message name="certificateServiceFault">
    <wsdl:part name="certificateServiceFault" element="tns:certificateServiceFaultElement"/>
  </wsdl:message>
- <wsdl:message name="getCertificateRequest">
    <wsdl:part name="getCertificatein" element="tns:getCertificatein"/>
  </wsdl:message>
- <wsdl:message name="getCertificateResponse">
    <wsdl:part name="getCertificateout" element="tns:getCertificateout"/>
  </wsdl:message>
```

Kuvio 9. Varmennepalvelun viestien määrittely

Elementillä voi olla ainoastaan name-attribuutti, joka kertoo viestin nimen. Lapsielementteinä voi olla ainoastaan part-elementti, jolla taas voi olla ainoastaan kolme erilaista attribuuttia. Yleensä attribuutteja on kuitenkin vain kaksi. Varmennepalvelun viestit ovat yksiosaisia, joten niillä on vain yksi part-lapsielementti. Attribuutilla element viitataan types-lohkossa määriteltyyn tietotyyppiin. [7; 9.]



Kuvio 10. Määritellyt tietotyypit liitetään välitettävään viestiin

3.1.3 Porttityyppi – portType-elementti

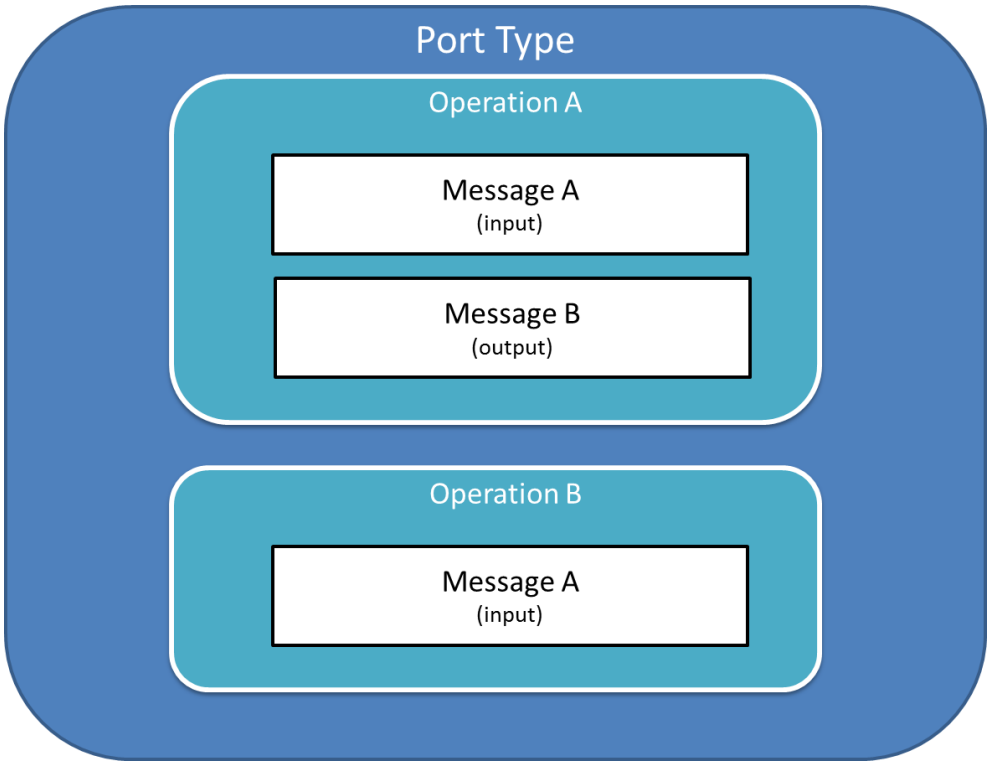
WSDL-dokumenteissa porttityypillä tarkoitetaan jonkin päätepisteen, joko käyttäjän tai palvelimen, tukemaa operaatiojoukkoa. Jokaisella porttityypillä voi olla yksi tai useampi operaatio. Operaatiossa puolestaan määritellään operaatiossa välitettävät aikaisemmin määritellyt viestit ja viestin kulkusuunta. Operaation attribuutilla name, annetaan operaatiolle nimi, jonka on oltava uniikki porttityypin sisällä, mutta samannimisiä operaatioita voi löytyä muista porttityypeistä. Operaatio elementin lapsielementteinä voi esiintyä input-, output- tai fault- elementit. Input ja output elementtien esiintyminen ja järjestys määrittää, minkä tyyppisestä operaatiosta on kyse. WSDL-standardi määrittelee neljä erityyppistä tiedonvälitystapaa.

- **yksisuuntainen** (one-way): päätepiste vastaanottaa viestin, ei vastaus viestiä, operaatio sisältää vain input-elementin.
- **pyyntö-vastaus** (request-response): Päätepiste vastaanottaa viestin ja palauttaa vastauksen, operaatiossa määritellään ensin input-elementti ja sitten output-elementti.
- **anottu vastaus** (solicit-response): päätepiste lähettää viestin ja saa vastauksen, operaatiossa määritellään ensin output-elementti ja sitten input-elementti.
- **ilmoitus** (notification): päätepiste lähettää viestin, ei vastaus viestiä, operaatio sisältää vain output-elementin.

```
- <wsdl:portType name="OPCertificateServicePortType">
    - <wsdl:operation name="getCertificate">
        <wsdl:input name="getCertificateRequest" message="tns:getCertificateRequest"/>
        <wsdl:output name="getCertificateResponse" message="tns:getCertificateResponse"/>
        <wsdl:fault name="certificateServiceFault" message="tns:certificateServiceFault"/>
      </wsdl:operation>
  </wsdl:portType>
```

Kuvio 11. Varmennepalvelun porttityypin määrittely

Input-, output- ja fault-elementtien message attribuutilla viitataan vastaanotettavaa tai vastaanotettavaan viestiin kuten esimerkissä operaatiossa getCertificate-palveluun vastaanotettava viesti on tns:getCertificateRequest-tyyppinen ja vastausviestinä on tns:getCertificateResponse-viesti. Fault-elemetillä kerrotaan, millainen vastausviesti palautetaan, mikäli vastaanotettua viestiä ei voitu käsitellä tai ei havaita oikeelliseksi. Fault-elemtille tulee aina määritellä nimi name-attribuutilla. Input- ja output-elementien nimeäminen ei ole pakollista, eli name-attribuutti on näille elementeille valinnainen. [7; 9.]



Kuvio 12. Porttityyppi määrittelee päätepisteen operaatiot ja operaatioissa välitettävien viestien kulkusuunnan.

3.1.4 Sidokset – binding-elementti

Sidosten tehtävä on määritellä tiedonsiirrossa käytettävän viestin muoto ja protokolla. Sidoselementillä binding on aina oma nimi, joka määritellään name-attribuutilla ja pakollinen viittaus johonkin määriteltyyn porttityyppiin. Porttityyppiin viitataan type-attribuutilla. Kuviossa 13. on kuvattu osa varmennepalvelun sidoksista ja operaatioista, binding ja operation.

```
- <wsdl:binding name="OPCertificateServiceHttpBinding" type="tns:OPCertificateServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  - <wsdl:operation name="getCertificate">
      <soap:operation soapAction=""/>
    - <wsdl:input name="getCertificateRequest">
        <soap:body use="literal"/>
      </wsdl:input>
    - <wsdl:output name="getCertificateResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    - <wsdl:fault name="certificateServiceFault">
        <soap:fault use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
```

Kuvio 13. Varmennepalvelun sidoksien määrittely

Binding-elementin sisällä olevan operation-elementin tulee loogisesti vastata sitä porttityypin määritystä operaatiolle, johon binding-elementin type-attribuutti viittaa. Sidoslohkossa WSDL-standardissa on määritelty viisi kohtaa , johon voidaan lisätä laajennuselementtejä. Laajennuselementtien avulla tehdään sidonta tiedonsiirtoprotokollaan ja viestin formaattiin. WSDL-versio 1.1 määrittelee sidonnat SOAP-, HTTP- sekä MIME-protokolliin, mutta tässä työssä tarkastelemme vain sidonnan toteutusta SOAP-protokollaan. SOAP-sidonnassa käytetään laajennuselementtejä jokaisessa viidessä WSDL:n määrittelemässä kohdassa eli binding-, operation-, input-, output- ja fault-elementtien jälkeen. SOAP-sidontaan tarvittavien laajennuselementien alussa on tunnus soap, joka viittaa elementtien käyttämään nimiavaruuteen. Laajennuselementit siis ovat eri nimiavaruudessa kuin varsinaisen WSDL-standardin elementit. Kuten wsdl-dokumentin alusta voimme lukea, nimiavaruuden tunnuksella soap viittataan URI-osoitteeseen http://schemas.xmlsoap.org/wsdl/soap, ja nimiavaruuden tunnus wsdl viittaa osoitteeseen http://schemas.xmlsoap.org/wsdl/. SOAP 1.1-sidonta on määritelty WSDL1.1-standardin luvussa 3.

Ensimmäinen sidontaan käytetty elementti on soap:binding, jonka attribuuteilla määritellään SOAP-tyyli ja viestien välitykseen käytettävä protokolla. Style-attribuutin arvolla document viitataan SOAP:in dokumentti tyyliseen sanomaan ja attribuutti transport määrittää protokollan, joka tässä tapauksessa on http, jolloin attribuutille annetaan arvoksi http://schemas.xmlsoap.org/soap/http. Soap:binding-elementti on aina pakollinen wsdl-dokumentissa, kun viestinvälitykseen käytetään SOAP-protokollaa.

Soap:operation-elementti ja sen attribuutti soapAction ovat pakollisia ainoastaan silloin, kun SOAP:n kanssa käytetään http-protokollaa. Attribuutin arvo voi olla mitä tahansa ja vastaa suoraan SOAPAction-otsikkoriviä http-pyynnössä.

Kolmas ja neljäs laajennuselementti on soap:body. Elementtien tarkoituksena on määrittää, kuinka viestien osat tallennetaan SOAP-viestin runko-osaan body. Elementin attribuuttina esimerkissä näemme attribuutin use, jonka arvona on literal. Tämä tarkoittaa, että SOAP-viestin body osassa välitettävää sisältöä ei enkoodata. Mikäli sisältö on enkoodattua attribuutin arvoksi tulee encoded. Viidentenä laajennuselementtinä on soap:fault. Tällä elementillä voidaan vaikuttaa siihen, kuinka SOAP-virheilmoitus muodostetaan siinä tapauksessa, ettei SOAP-sovellus ymmärtänyt lähettäjän pyyntöä tai ettei pyyntöä voitu toteuttaa. [7; 8; 9]

3.1.5 Service- ja port-elementit

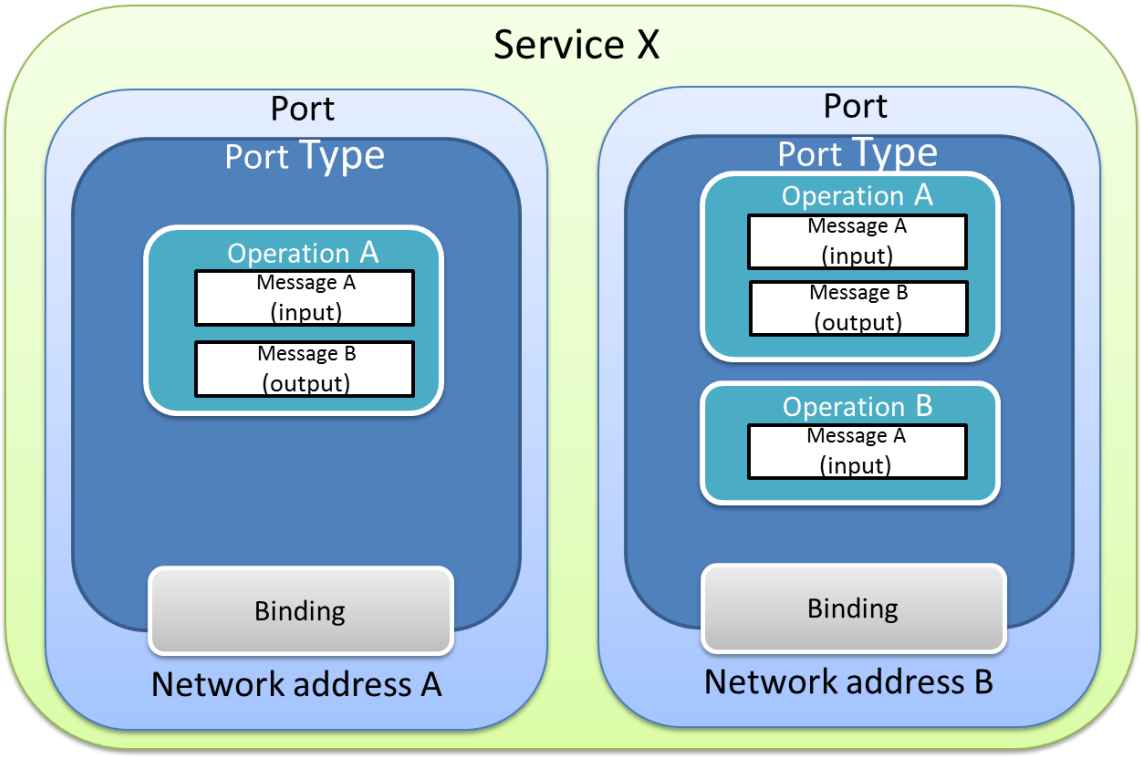
WSDL:n service-elementin tarkoitus on koota yhteenkuuluvat portit yhdeksi palveluksi. WSDL-dokumentissa voidaan määritellä useitakin erilaisia palveluita, mutta yleensä palvelumäärityksiä on kuitenkin vain yksi.

Service-elementin sisällä oleva port-lapsielementti viittaa binding-attribuutilla johonkin määriteltyyn sidokseen. Kun käytetään WSDL:n SOAP-sidontaa, port-elementin tulee sisältää yksi soap:address-elementti. Soap:address-elementin location-attribuutilla määritetään se URI-osoite, jolla SOAP-palvelua voidaan kutsua. Kuviossa 14 on kuvattu varmennepalvelun määrittelyn service- ja port-elementit.

```
- <wsdl:service name="OPCertificateService">
    - <wsdl:port name="OPCertificateServiceHttpPort" binding="tns:OPCertificateServiceHttpBinding">
        <soap:address location="http://domain.fi:1001/wsdl/CertificateService.xml"/>
      </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Kuvio 14. Varmennepalvelun service- ja port-elementtien määrittelyt

Yksi palvelu voi sisältää useita verkko-osoitteita, eli yhden service-elementin sisään voi olla määriteltynä useampi port-elementti. Port-elementit voivat viitata samaankin sidokseen, jolloin ne tarjoavat ikään kuin vaihtoehtoisen osoitteen samalle operaatiolle. Kuvassa 15 on havainnollistettu laatikkomallilla palvelua, jolla on kaksi toiminto kokonaisuutta, omissa osoitteissaan. [7; 8; 9.]



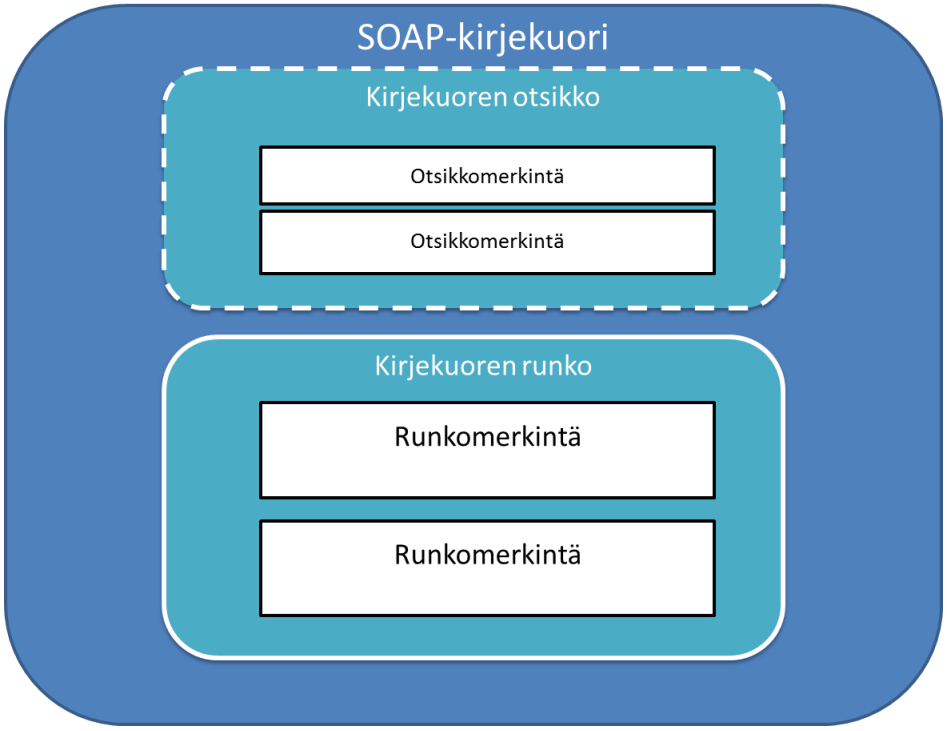
Kuvio 15. Port-elementeissä määritettään porttityypille ja sen operaatioille verkko-osoite, service niputtaa kaikki wsdl-dokumentissa kuvatut toiminnot palveluksi.

3.2 SOAP Simple Object Access Protocol

SOAP-määritys koostuu neljästä osasta: SOAP-kirjekuoren (envelope), koodauksen (encoding), etäproseduurikutsuista (RPC) ja SOAP:in http-sidonnasta. Pankkiyhteysliikenteessä kaikki SOAP-viestit ovat tyypiltään dokumentti tyylisiä, jonka vuoksi tässä työssä ei ole tarkasteltu etäproseduurikutsujen käyttöä. [9; 10.]

3.2.1 SOAP-kirjekuori

SOAP-viestejä kutsutaan myös kirjekuoriksi (envelope). Kirjekuori on aina XML-muotoinen dokumentti, jolla ei saa olla xml:n prosessointikäskyjä, mutta toisaalta kirjekuori voi alkaa XML-julistuksella. SOAP-kirjekuori alkaa aina Envelope-elemetillä ja nimiavaruutena käytetään http://schemas.xmlsoap.org/soap/envelope. SOAP-kirjekuoren encodingStyle-attribuutin arvona on yleensä http://schemas.xmlsoap.org/soap/encoding. Soap-kirjekuori pitää sisällään pakollisen runko-osan eli Body-elementin ja valinnaisen Header-otsikkoelementin. [9; 10.]



Kuvio 16. Kuviossa on kuvattu SOAP-kirjekuoren rakenne.

Käytännön esimerkki downloadFilelist-toiminnon SOAP-sanomasta. Sanoman Header-osasta on paremman luettavuuden vuoksi piilotettu sanoman digitaalinen allekirjoitus sekä valtaosa kirjekuoren Body-elementissä olevan ApplicationRequest-elementin Base64-koodatusta sisällöstä on poistettu:

```
<?xml version="1.0" encoding="UTF-8"?>
- <SOAP-ENV:Envelope xmlns:ns2="http://bxd.fi/CorporateFileService"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns1="http://model.bxd.fi" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  - <SOAP-ENV:Header>
      + <wsse:Security SOAP-ENV:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
        open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body wsu:Id="pfx2ae2e374-edde-f219-43a4-b631299f847e" xmlns:wsu="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      - <ns2:downloadFileListin>
          - <ns1:RequestHeader>
              <ns1:SenderId>17179681</ns1:SenderId>
              <ns1:RequestId>1395396506</ns1:RequestId>
              <ns1:Timestamp>2014-03-21T12:08:26+02:00</ns1:Timestamp>
              <ns1:Language>FI</ns1:Language>
              <ns1:UserAgent xsi:nil="true"/>
              <ns1:ReceiverId/>
            </ns1:RequestHeader>
            <ns1:ApplicationRequest>PD94bzdD4K</ns1:ApplicationRequest>
        </ns2:downloadFileListin>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Kuvio 17. Käytännön esimerkki SOAP-kirjekuoresta, jota on muokattu helpommin luettavaan muotoon.

Runko-osassa eli Body-elementti ja sen lapsielementit sisältävät tiedon toiminnosta, lähettäjän tunnuksen sekä applicationRequest-elementissä Base64-koodattuna välitettävästä xml-dokumentista. [2.]

3.2.2 Header-elementti

SOAP-kirjekuoren otsikko alkaa aina Header-elementillä. Standardissa ei määritellä otsikolle mitään pakollista sisältöä, vaan kaikki otsikossa olevat tiedot määritellään sovelluskohtaisesti. Jokaista Header-elementin lapsielementtiä kutsutaan otsikkomerkinnäksi. Otsikkomerkinnöissä käytetään usein laajennuselementtejä, jotka eivät kuulu SOAP-standardiin. Tässä esimerkissä otsikkomerkintänä on Security-elementti, jonka nimiavaruus viittaa OASIS-työryhmän määrittelemään WS Security-standardiin.

Vaikka otsikkomerkinnät eivät kuulu SOAP-määritykseen, ensimmäisen tason otsikkomerkinnöissä voidaan käyttää kolmea SOAP-määritykseen kuuluvaa attribuuttia.

- actor
- encodingStyle
- mustUnderstand.

EncodinStyle-attribuutilla määritellään, kuinka tietoja tulisi välittää lankaa pitkin. Actor-attribuutin tarkoituksena on kertoa, mille sovellukselle kyseinen otsikkomerkintä on tarkoitettu. Koska SOAP-sanoma voi kulkea useammankin sovelluksen kautta matkalla lähettäjältä vastaanottajalle, voi osa otsikkomerkinnöistä olla tarkoitettu näille välittäjäsovelluksille (SOAP intermediary). Kun mahdollinen välittäjä tunnistaa itsensä actor-attribuutista, tulee sovelluksen poistaa nämä otsikkomerkinnät ennen viestin eteenpäin välittämistä. Ehkä yleisimmin törmää kolmanteen, DownloadFileList-esimerkissäkin käytettyyn mustUnderstand-attribuuttiin. Tällä attribuutilla määritellään ikään kuin otsikkotiedon pakollisuus. Attribuuttille sallitut arvot ovat ”0” j a”1”. Ensimmäistä kuitenkin näkee aika harvoin sillä arvo 0, nolla, tarkoittaa käytännössä sovellukselle samaa kuin attribuuttia ei olisi ollenkaan. Arvolla 1, yksi, otsikkomerkintää käsittelevän sovelluksen on ymmärrettävä kyseisen otsikkomerkinnän merkitys. Mikäli näin ei ole tulee sovelluksen aiheuttaa virhetilanne ja palauttaa SOAP-viestin lähettäjälle virheilmoitus. [9; 10; 11.]

3.2.3 SOAP runko eli Body

Runko-osa on aina pakollinen osa SOAP-viestiä. Body-elementin sisällä välitetään halutunlaista XML-muotoista tietoa, RPC-kutsun parametreja tai virheilmoituksia. SOAP-standardi ei määrittele mitä Body-elementti saa sisältää. Tämän vuoksi kaikki runko-osassa esiintyvät elementit eli runkomerkinnät tulisi määritellä täydellisillä nimillä, eli jokaiselle, myös lapsielementeille, on määritetty nimiavaruus. Toisin kuin kirjekuoren otsikossa, vastaanottajan tulee aina tunnistaa runko-osassa olevien elementtien merkitys, vaikkei elementeille määritelläkään mustUnderstand-attribuuttia. [9; 10.]

3.2.4 Virheiden käsittely ja Fault-elementti

SOAP-viestien käsittelyssä voi tulla odottamattomia virheitä ja niistä välitetään tieto viestin lähettäjälle SOAP-vastausken runko-osassa olevan fault-elementin avulla. SOAP-standardi määrittelee Fault-elementille neljä lapsielementtiä.

- faultcode (pakollinen): Elementissä on tarkoitus välittää konekielinen syy virheentunnistamiseen.
- falutstring (pakollinen): Elementissä välitetään inhimillinen versio virheilmoituksesta esimerkiksi ”File not Found”.
- faultfactor (valinnainen): Elementissä kerrotaan SOAP-virheen aiheuttaneen sovelluksen tiedot. Arvona palautetaan URI-osoite hieman samaan tapaan kuin SOAP-otsikossa käytettävä actor-attribuutti.
- detail (pakollinen/valinnainen): Tässä elementissä välitetään tarkempi kuvaus virhetilanteesta ja sillä voi olla myös lapsielementtejä. Jos SOAP-viestin runko-osan käsittely aiheutti virhetilanteen, tämä elementti on pakollinen. Toisaalta tätä elementtiä ei saa käyttää sellaisen virheen kuvaamiseen, joka on aiheutunut otsikkomerkintöjen käsittelystä. [9; 10.]

3.2.5 SOAP:in http-sidonta

SOAP-standardi määrittelee tavan, jolla SOAP-viestejä voidaan välittää http-protokollan päällä. Vaikka SOAP onkin neutraali käytetyn tiedonvälitysprotokollan suhteen, valtaosa SOAP-viestejä hyödyntävistä sovelluksista on toteutettu juuri http:n päällä, sillä http:n pyyntö-/vastausarkkitehtuuri noudattaa SOAP:n sanoma-arkkitehtuuria.

HTTP-protokolla on suunniteltu laajennettavaksi HTTP-pyynnön otsikkokenttiä hyödyntäen. SOAP:n http-sidonta toteutetaan käyttämällä SOAPAction-kenttää, jolla voidaan tarkentaa SOAP-kutsun tarkoitusta, mutta sillä ei ole pakko olla mitään arvoa. Uudemmassa SOAP-versiossa 1.2 ei SOAPAction kentän käyttö ole enää pakollista, sillä tieto HTTP-sanoman sisältämästä SOAP-kutsusta ilmenee ContentType-kentästä.

SOAPAction-kentän sisältö määräytyy WSDL-dokumentin perusteella. WSDL:n binding-elementin elementin alta löytyvän SOAP-sidonnan mukainen laajennuselementti soap:operation, jonka atribuuttina löytyy soapAction.

SOAP-viestien välittämiseen käytetään aina standardin mukaisesti HTTP-protokollan POST-metodia. HTTP:llä välitetyn kutsun sisältötyyppi (Content-Type) SOAP-Versiossa 1.1 on text/xml ja versiossa 1.2 käytetään arvoa application/soap. SOAP-version 1.1 kutsu välitettynä HTTP-protokollan päällä voisi näyttää kuvion 18.mukaiselta.

```
POST /StockQuote HTTP/1.1
Content-Type: text/xml; charset="utf-8"
Content-Length: 9052
SOAPAction: ""

<SOAP-ENV:Envelope
...
```

Kuvio 18. Esimerkki SOAP-pyynnöstä HTTP-protokollaa käyttäyttäen

SOAP-kutsun paluuarvot välitetään takaisin kutsun tehnelle sovellukselle HTTP-vastauksen mukana. Jos SOAP kutsu onnistui, HTTP:n paluuarvo koodin on oltava 200 sarjan koodi, esimerksiki ”200 OK”. Jos SOAP-kutsua ei voitu käsitellä oikein, on paluukoodin oltava ”500 Internal Server Error”. Virhetilanteessa käyttäjälle palautettavassa SOAP-kirjekuoressa on oltava mukana Fault-elementti. [9; 10.]

4 Web Sercives pankkiyhteyskäytössä

Pankkiyhteyskäytössä Web Services -palvelu koostuu kahdesta erillisestä SOAP-sanomiin pohjautuvasta palvelusta: pankkiaineistojen siirrossa käytettävästä aineisto-

siirtopalvelusta varmenteiden noutoon, uusintaan ja hallintaan tarkoitetusta varmenne palvelusta.

Suomessa web services -kanavan määrittelystä vastannut työryhmä koostui Nordean, OP-Pohjola Groupin ja Sampo-pankin (nykyinen Danske Bank) asiantuntijoista. Määrittely sisältää aineistosiirtopalvelun WSDL-dokumentaation ja sekä sanomien sisältämän ApplicationRequest ja ApplicationResponse XML-dokumenttien schemat. Varmennepalvelun periaatteet määritellään dokumentaatiossa yleisellä tasolla, mutta teknisen pankin tai sen ratkaisutoimittajan tehtäväksi jätettiin:

- PKI määrittely etenkin, kuinka varmenteet toimitetaan käyttäjälle
- yhteyden testauspalvelut
- listan tuetuista aineistotyypeistä ja operaatioista
- muut pankkikohtaiset ohjeistukset joihin määrittelyssä ei oteta kantaa esimerkiksi palveluiden verkko-osoitteet ja niin edelleen.

Määrittelyssä uudelle yhteystavalle asetettiin seuraavia vaatimuksia, jotka Web Services yhteyskäytännön tulisi täyttää:

Yhteyden tulee perustua kansainvälisiin standardeihin, jotta sovelluskehitykseen voidaan käyttää moderneja sovelluskehitystyökaluja. Kansainvälisiin standardeihin pohjautuva ratkaisu rohkaisee kotimaisten ja kansainvälisten ohjelmistotoimittajien tarjoamaan valmiita komponentteja ja ohjelmistoratkaisuja yritysten maksuliikenteen hoitoon.

- Pankki voi itsenäisesti valita luottamansa varmenteiden myöntäjän.
- Yhteyden tulee käyttää salaus algoritmejä ja avainpituuksia, jotka tarjoavat finanssialalla vaaditun turvallisuustason vähintään kahdeksikymmeneksi vuodeksi.
- Tieliikenneyhteyden turvallisuuden erottaminen itse siirrettävästä pankkidatasta, jolloin pankin yritysasiakas voi ulkoistaa aineiston muodostamisen ja/tai web services -yhteyden palvelukeskukselle tai muulle operaattorille.
- Yhteystietoturvan rajoittaminen vain pankin ja yhteydessä olevan asiakkaan välille, oli kyseessä sitten pankin loppuasiakas, palvelukeskus, tai muu aineiston toimittaja. Tämä tarkoittaa että päästä-päähän-salausta (end-to-end encryption) ja web services -sanoman salausta ei vaadita, koska SSL tai VPN varmistavat siirtokanavan turvallisuuden julkisessa verkossa.

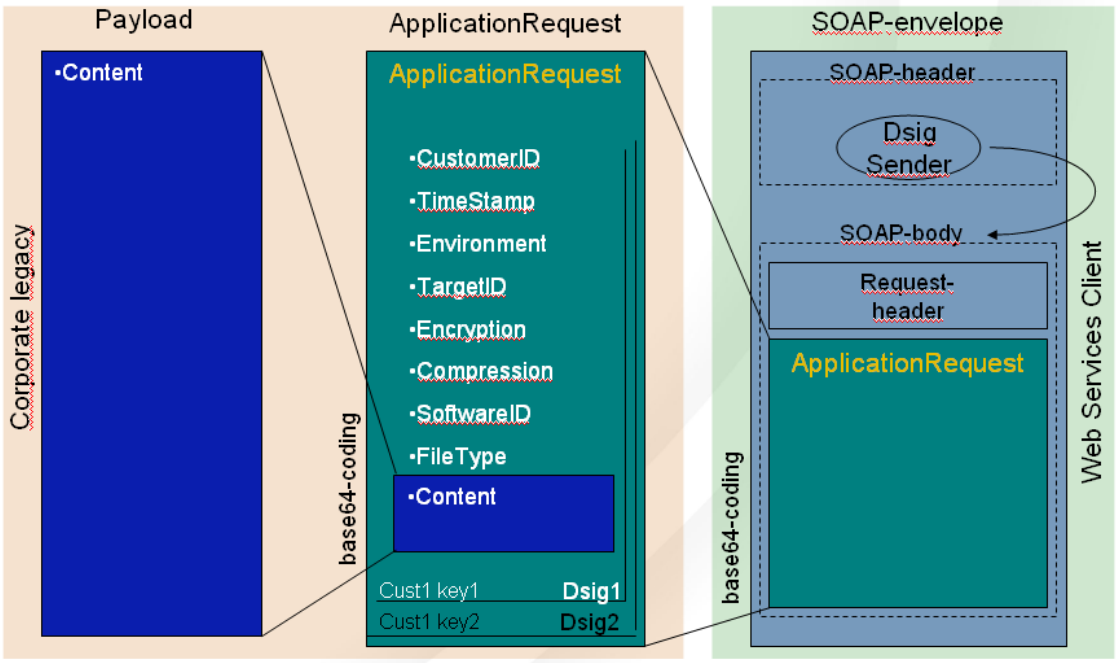
- .Valinnainen siirrettävän sisällön salaus tullaan määrittelemään, mutta XML Encryption ei ole Web Services -tason salausmekanismi, eikä se perustu Web Services -standardeihin. XML Encryption on XML-standardi ja käytetään varsinaisen sisällön salaamiseen, ei web services -sanoman salaukseen. XML-salaus ja salauksen purku tapahtuu itse sanoman sisältöä käsittelevissä ohjelmistoissa, ei Web Services -asiakasohjelmistossa tai palvelin sovelluksessa. [2.]

4.1 Sanoman muodostus

Määrittelyyn mukaan haluttiin tarjota mahdollisuus, jossa pankin asiakkaan on mahdollista ulkoistaa aineistopalvelut ulkoiselle palveluntarjoajalle osittain tai kokonaan. Tämä luo pankkiin lähetettävälle sanomalle suhteellisen monimutkaisen rakenteen ja syvän hierarkian, joka sisältää hyötykuorman (Payload), sovellupyynnön (ApplicationRequest) ja vielä itse välitettävän SOAP-kirjekuoren. Tämän lisäksi sanoma digitaalisesti allekirjoitetaan kahdessa eri vaiheessa, kun käyttöön on varattu aineiston allekirjoittajalle (ApplicationRequest) ja aineiston lähettäjälle (SOAP) mahdollisuus tunnistautua omilla allekirjoituksillaan. Alla on listattuna vaiheittain yksinkertaistettuna SEPA-maksuaineisto sanoman muodostus ja lähetys pankin Web Services -palvelun kautta käsiteltäväksi.

1. Luodaan käsittelyyn haluttavien maksujen tietojen pohjalta ISO20022-standardin mukainen maksuaineisto eli hyötykuorma.
2. Hyötykuorma Base64-koodataan.
3. Luodaan xml-dokumentti nimeltään ApplicationRequest eli sovelluspyyntö. Sovelluspyyntö sisältää muun muassa tiedot mitä hyötykuorma sisältää, jotta pankin päässä se osataan ohjata oikeaan paikkaan prosessoitavaksi.
4. Base64-koodattu hyötykuorma sijoitetaan ApplicationRequestin Content-elementtiin.
5. Koko ApplicationRequest-dokumentti allekirjoitetaan allekirjoitusvarmenteen yksityisellä avaimella XML-standardin mukaisesti (11.)
6. Allekirjoituksen jälkeen ApplicationRequest Base64-koodataan.
7. Luodaan WSDL:ssä määritelty UploadFile-toiminnon mukainen SOAP-sanoma.
8. Sijoitetaan allekirjoitettu ja Base64-koodattu ApllicationRequest SOAP-sanoman runko-osaan.

9. Digitaalisesti allekirjoitetaan SOAP-kirjekuori sanoman otsikko-osaan käyttäen nyt lähettäjän varmenteen yksityistä avainta. (12.)
10. Lähetetään allekirjoitettu WSDL:ssä kuvattuun verkko-osoitteeseen ja odotetaan SOAP-vastausta. [2.]



Kuvio 19. Havainne kuva sanoman muodostuksesta [1.]

4.2 Kaksitasoinen todentaminen

Kuvatunlainen kaksi tasoinen todentaminen, jossa sanoma allekirjoitetaan kahdessa vaiheessa, luo käyttäjän näkökulmasta varmenteiden haltijoille omat roolinsa.

Allekirjoitusvarmenteen haltija vastaa liiketoimintakriittisen datan oikeellisuudesta esimerkiksi tilinumerot, maksujen saajan, summat ja niin edelleen. Tämän allekirjoituksen perusteella voidaan todentaa, että henkilö tai osasto, joka hyötykuorman on luonut ja allekirjoittanut, on oikeutettu myös käyttämään esimerkiksi maksamaan maksuja yrityksen tililtä.

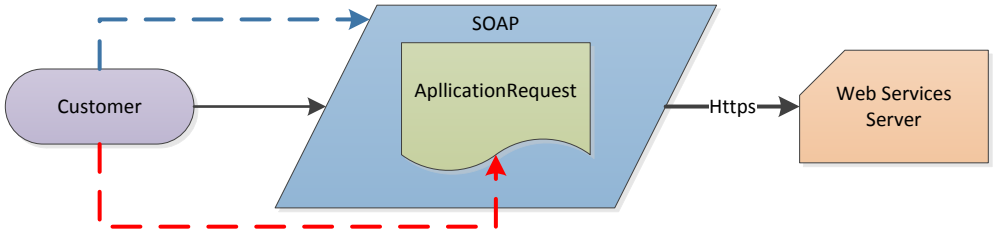
Lähettäjänvarmenteen haltijan rooli on teknisempi , tämän osapuolen tarkoitus on vastata pankkiyhteyden teknisestä toimivuudesta. Yksi suuri syy myös lähettäjän tasolla tehtävään tunnistukseen on estää tai pienentää DoS – Denial-of-Service-hyökkäyksen riskiä ja onnistumista. Mikäli sanoman lähettäjää ei tunnisteta heti SOAP-sanoman

allekirjoituksesta, voidaan tämän lähettäjän sanoma hylätä ilman, että aineiston prosessointiin olisi kulutettu hirvesti resursseja pankin järjestelmässä. [2.]

4.3 Erilaiset kokoonpanoratkaisut

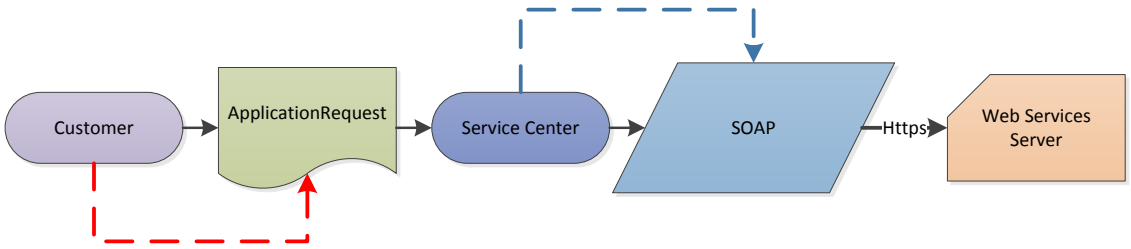
Seuraavassa tutkitaan neljää mahdollista järjestelyä, miten sanoman muodostus, allekirjoitukset ja lähetys voitaisiin toteuttaa käyttäjän näkökulmasta. Kuvioissa punaisella katkoviivalla on kuvattu allekirjoitusvarmenteen haltija ja sinisellä katkoviivanuolella lähettäjänvarmenteen haltijaa. Kuviossa 19 on kuvattuna kaikkein yksinkertaisin toteutus, jossa pankin asiakas suorittaa kaikki vaaditut toimenpiteet itse.

- Muodostaa hyötykuorman ja allekirjoittaa ApplicationRequestin.
- Muodostaa ja allekirjoittaa SOAP-sanoman.
- Lähettää SOAP-sanoman suojatun HTTPS-yhteyden yli.



Kuvio 20. Yksinkertaisin järjestely sanoman muodostukseen ja lähetykseen

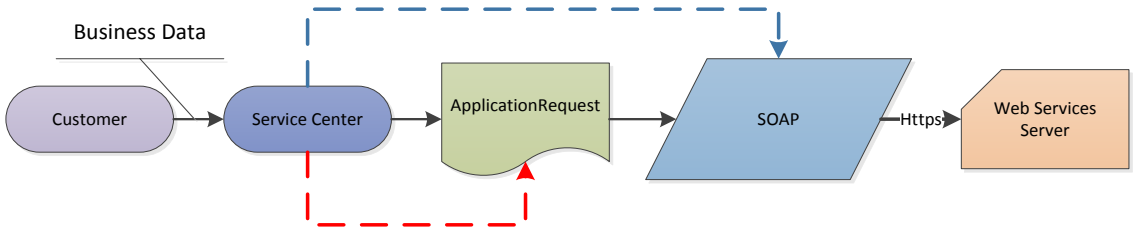
Toisessa kuvion 20 mukaisessa esimerkki tapauksessa pankin asiakas on ulkoistanut Web Services -rajapinnan kolmannelleosapuolelle, joka hoitaa asiakkaan puolesta SOAP-sanoman muodostuksen, allekirjoituksen ja lähetyksen.



Kuvio 21. Asiakas on ulkoistanut WS-rajapinnan palvelukeskukselle.

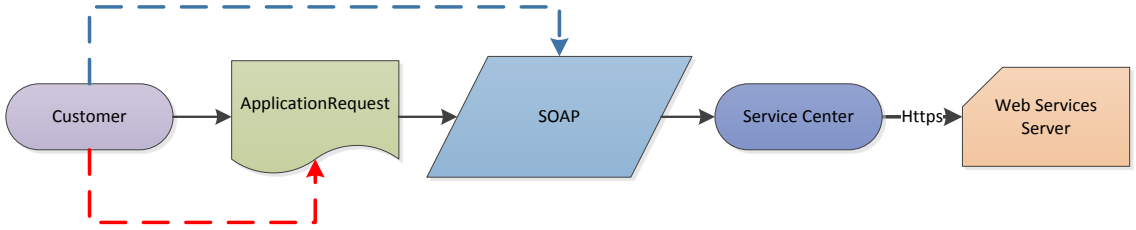
Kolmas järjestely voisi tulla vastaan tilanteessa, jossa pankin asiakkaalle ei ole käytössään ohjelmistoa, jolla hän voisi muodostaa esimerkiksi ISO20022-maksuaineistoa tai

Finvoice-standardin mukaista verkkolaskuaineistoa. Asiakas toimittaa palvelukeskukselle tarvittavat tiedot niin sanottuna ”raakadatana”, josta palvelukeskus muodostaa pankinjärjestelmien mukaisen hyötykuorman, sekä hoitaa kaikki tarvittavat vaiheet pankin WS-rajapintaan saakka.



Kuvio 22. Palvelukeskus hoitaa aineiston- muodostuksen, allekirjoituksen ja lähetyksen asiakkaan antamien tietojen perusteella.

Neljännessä skenaariossa pankin asiakas hoitaa itse ApplicationRequestin sekä SOAP:in muodostukset ja allekirjoitukset, mutta on ulkoistanut vain salatun yhteyden julkisen verkon yli pankkiin. [2.]



Kuvio 23. Palvelukeskus hoitaa vain salatun yhteyden julkisen verkon yli.

4.4 Aineistosiirrossa käytettävien sanomien määrittelyt

4.4.1 SOAP Message

SOAP-sanoman rakenne on melko yksinkertainen, sillä kaikki WS-rajapinnasta varsinaiselle pankkisovelluksille välitettävä data on Base64-koodattuna ApplicationRequest-elementin sisään. SOAP-sanomassa välitetään haluttu operaatio, joita Samlinkin toteutuksessa on tuettuna kolme kappaletta:

- UploadFile – Aineiston lähetyspankkiin
- DownloadFileList – Operaatio palauttaa käyttäjälle listan pankista ladattavista tiedostoista ja niiden yksilöset tunnisteet

- DownloadFile – Operaatiolla ladataan yksittäinen tiedosto käyttäen lista kyselyllä saatua tiedostontunnistetta.
- DeleteFile – Operaatiolla voidaan peruuttaa lähetetty aineiston, mutta vain siinä tapauksessa ettei sitä ole ehditty käsittelemään pankin tietojärjestelmässä.

ApplicationRequest-elementin lisäksi SOAP-pyynnön runko-osassa on Request-Header-elementti, jossa välitetään sanoman parametreja sekä käyttäjän tunnistuksessa hyödynnettävää tietoa. RequestHeaderissä käytettävät tiedot on kerrottu pankkikohtaisissa sovellusohjeissa.

SOAP-sanoman otsikko-osaan tulee lähettäjän varmenteella muodostettu OASIS-standardin mukainen WS Security X.509v3 Certificate Token. [2;3;13.]

4.4.2 ApplicationRequest

Samlinkin teknisessä rajapintakuvauksessa ApplicationRequestin elementtien käytöstä on määritelty seuraavasti:

- CustomerId - Aineiston muodostajan yksilöivä tunniste eli palvelutunnus. Tämä on luovutettu yritykselle sopimuksen allekirjoituksen yhteydessä.
- Command - Arvon tulee vastata SOAP-operaatiota.
- Timestamp - Aikaleima jolloin ApplicationRequest on luotu
- StartDate – Käytetään DownloadFileList operaatiolla rajaamaan palautettavien ladattavien tiedostojen FileReference listaa tiedostojen luontiajan mukaan.
- EndDate - Käytetään DownloadFileList operaatiolla rajaamaan palautettavien ladattavien tiedostojen FileReference listaa tiedostojen luontiajan mukaan.
- Status - Käytetään DownloadFileList operaatiolla rajaamaan palautettavien ladattavien tiedostojen FileReference listaa tiedostojen tilan mukaan, koodit NEW, DLD, ALL.
- ServiceId - Ei käytössä.
- Environment – Tuettu arvot: PRODUCTION ja TEST. Testi-ympäristössä käytetään tuotannon tunnuksia, mutta aineistot eivät etene käsittelyyn.
- FileReferences/FileReference – Ladattavan tai poistettavan tiedoston ykslöivä tunniste. Tuettuna operaatioille: DownloadFile ja DeleteFile.
- UserFileName - elementin arvon ainoastaan operaatioille: UploadFile.
- TargetId - Tuettu määritelmän mukaisesti
- ExecutionSerial - Ei käytössä
- Encryption - Ei käytössä
- EncryptionMethod - Ei käytössä
- Compression - Ilmaistaan onko aineistossa käytetty pakkausta. Arvo boolean tyyppinen, joko true tai false.
- CompressionMethod - Jos aineistossa käytetään pakkausta, arvo ”GZIP”
- AmountTotal - Tuetaan operaatioille: UploadFile. ISO20022-maksuaineistolle verrataan elementin arvoa maksuaineiston summaan.

- TransactionCount - Tuetaan operaatioille: UploadFile. ISO20022-maksuaineistolle verrataan elementin arvoa maksuaineistossa olevien tapahtumien kappalemäärrään.
- SoftwareId - Asiakkaan sovelluksen lähettämä tieto ohjelmistosta ja sen versiosta.
- CustomerExtension - Ei käytössä.

Kuvio 24. Kuviossa ApplicationRequestin dokumentaatiokaavio [2.]

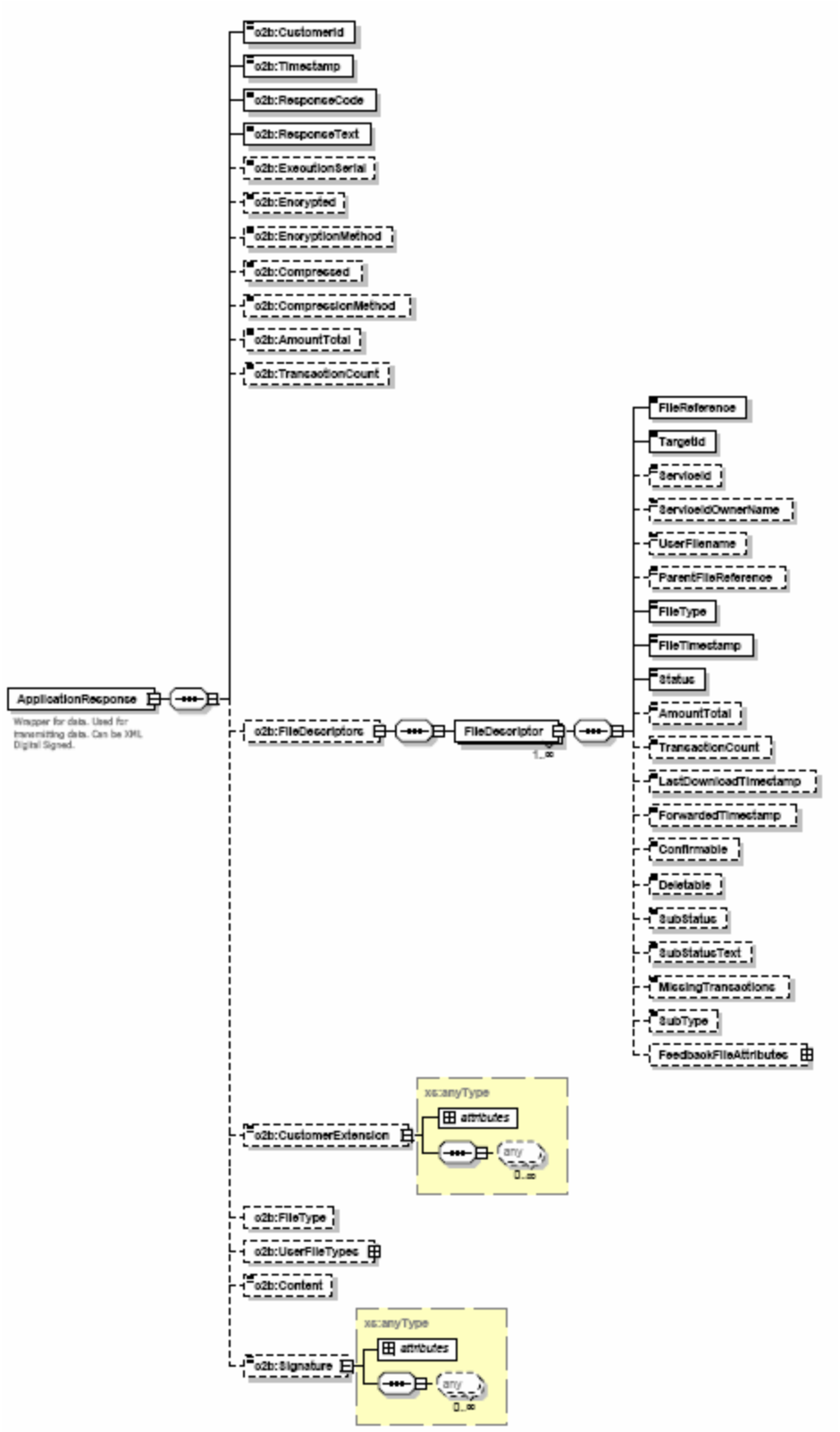
4.4.3 ApplicationResponse

Samlinkin teknisessä rajapintakuvauksessa ApplicationResponsen elementtien käytöstä on määritelty seuraavasti:

- CustomerId - Asiakkaan palvelutunnus.
- Timestamp -Aikaleima, jonka perusteella aineisto voidaan todeta vanhentuneeksi.
- ResponseCode - Katso taulukko 3.
- ResponseText - Katso taulukko 3.
- ExecutionSerial - Ei käytetä
- Encrypted - Ei käytetä
- EncryptionMethod - Ei käytetä
- Compressed - Ilmaistaan onko aineistossa käytetty pakkausta. Arvo boolean tyyppinen, joko true tai false.
- CompressionMethod - Tuemme RFC 1952 mukaisesti
- AmountTotal - Katso requestin selitys
- TransactionCount - Katso requestin selitys
- CustomerExtension -Ei käytössä
- FileDescriptors – Elementti palautetaan ainoastaan operaatiolla DownloadFileList
- FileDescriptor - Elementti palautetaan ainoastaan operaatiolla DownloadFileList
    - FileReference (aina)
    - TargetId (aina)
    - ServiceId (ei käytetä)
    - ServiceIdOwnerName (ei käytetä)
    - UserFileName (aina)
    - ParentFileReference (jos kyseessä palaute)
    - FileType (aina)
    - FileTimestamp (aina)
    - Status (aina)
    - AmountTotal (aina)
    - TransactionCount (aina)
    - LastDownloadTimestamp (annetaan palautteille jos ei tyhjä)
    - ForwardedTimestamp (palautetaan aineistolle jos ei tyhjä)
    - Confirmable (ei käytetä)
    - Deletable (ei käytetä)
    - SubStatusCode (ei käytetä)
    - SubStatusText (ei käytetä)
    - MissingTransactions (ei käytetä)
    - SubType (ei käytetä)
    - FeedbackFileAttributes (ei käytetä)

Taulukko 3. Taulukossa Samlink palauttamat virheilmoitukset ResponseCode- ja ResponseText-elenemteissä.

Koodi ResponseCode	**Selite** ResponseText
00	OK
05	Tuntematon sovelluspyyntö
12	Aineiston muodollinen tarkistus epäonnistui
21	Aineiston pakkaus viallinen
25	Aineisto ei ole sallittu
26	Tekninen virhe
27	Aineistoa ei voitu poistaa
30	Tunnistus epäonnistui
32	Kaksoislähetys



Kuvio 25. Kuviossa ApplicationResponsen dokumentaatiokaavio [2.]

5 Varmennepalvelu ja PKI

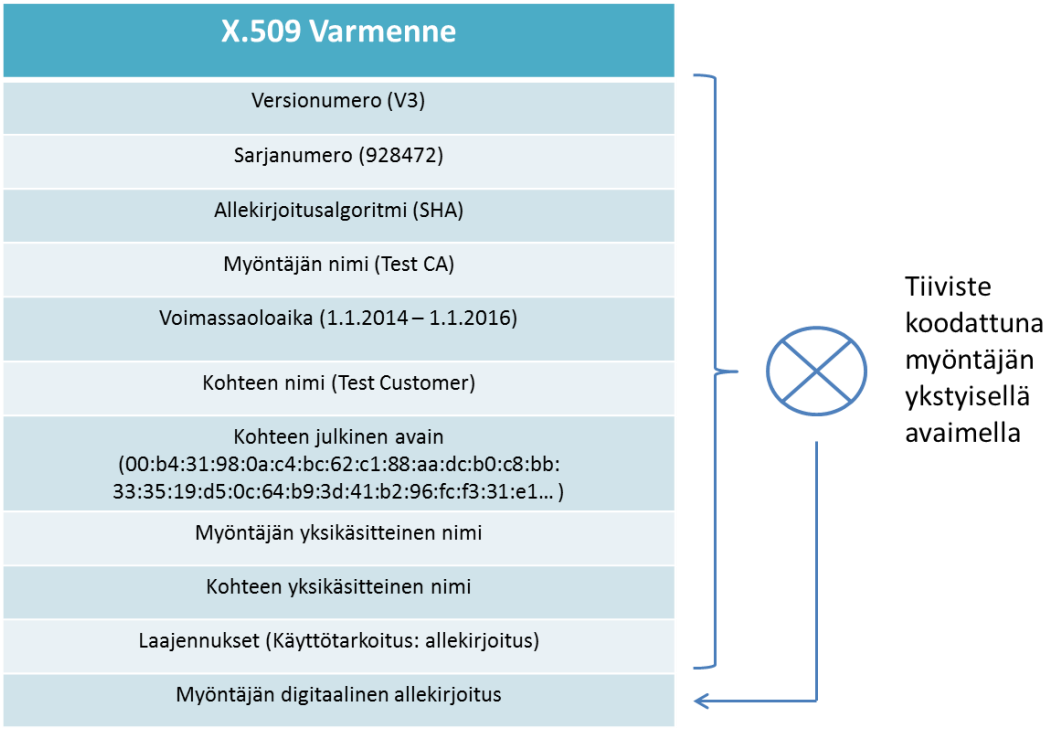
Web Services -sanomien muodostuksen yhteydessä puhuttiin paljon sanomien digitaalisesti allekirjoittamisesta varmenteella. Digitaalinen allekirjoitus perustuu asymmetriseen salaustekniikkaan, jossa on käytössä kaksi salausavainta, julkinen ja yksityinen avain. Yksityisellä avaimella salattu tieto voidaan avata selkokieliseksi julkisella avaimella. Julkinen avain voidaan täten jakaa kenen tahansa saataville. Julkisen avaimen järjestelmä on kuitenkin vain tekniikka, josta ei yksinään ole paljoakaan hyötyä, mikäli ei ole varmuutta yksityisen avaimen haltijasta. Varmenne eli sertifikaatti onkin sähköinen todistus, jonka varmenteen myöntäjä eli CA (Certificate Authority) on sähköisesti allekirjoittanut. Varmenteen hakijan on ensin muodostettava käyttöönsä yksityinen ja julkinen avain. Varmenteen hakija toimittaa julkisenavaimensa ja CA:n vaatimat tiedot yksityisen avaimen haltijasta varmenteen myöntäjälle. Varmenteen myöntäjä tarkastaa varmennepyynnössä olevat tiedot. CA:n todettua hakijan lähettämien tietojen oikeellisuuden myöntäjä laskee varmennepyynnössä olevista tiedoista tiivisteen ja allekirjoittaa varmenteen digitaalisesti. Sovellus, jolle varmenne esitetään, tarvitsee CA:n julkisen avaimen purkaakseen tiivisteen ja verratakseen sitä itse laskemaansa. Jos tulokset täsmäävät, varmenteessa kerrottua tietoa voidaan pitää uskottavana.

Varmenteen kertomiin tietoihin voidaan luottaa, mikäli seuraavat ehdot täyttyvät:

- Varmeenteen myötäjä on luotettava taho.
- Varmenteen myöntäjän yksityisen avaimen on pysyttävä salassa.
- Varmenteen myöntäjän julkinen avain on saatu turvallista kanavaa pitkin.
- Varmenteessa käytetty salaustekniikka on riittävän turvallinen (algoritmit, avainpituudet). [14.]

5.1 X.509-standardi

Standardoidulla varmenteiden esitystavalla pyritään varmistamaan, että eri tahojen myöntämät varmenteet olisivat keskenään yhteensopivia. International Telecommunication Union (ITU) on määritellyt varmenteiden esitystapaa varten X.509-standardin. [14.]



Kuvio 26. X.509-standardi määrittelee varmenteen tietokentät ja niiden esitystavan

Kuten aiemmin mainittuna myönnetyn varmenteen käyttöä SOAP-sanoman allekirjoitukseen ohjaa Web Services Security X.509 Certificate Token Profile 1.1-standardi ja xml-dokumenttien allekirjoitus on määritelty W3C:n XML Signature Syntax and Processing-standardissa [11; 12.]

5.2 PKI-järjestelmä

PKI-järjestelmäksi (Public Key Infrastructure) kutsutaan varmenteiden myöntämiseen, jakeluun ja ylläpitoon kuuluvaa kokonaisuutta. Kyseiset tukitoiminnot tuovat varmenteiden käyttöön helppoutta, kattavaa ja turvallista. PKI-järjestelmän keskeisimmässä roolissa toimii varmenteen myöntäjä (Cetficate Authority, CA). Periaatteessa CA voisi toimittaa varmenneprosessin kaikkia rooleja aina avainten muodostuksesta varmenteen jakeluun ja ylläpidollisiin tehtäviin. Usein etenkin kuitenkin vastuu avainparin muodostuksesta etenkin niin sanottujen software- eli ohjelmallisesti käytettävien kohdalla on varmenteen hakijalla. Hakijan tietojen tarkastus hoidetaan usein erillisen rekisteröijän (Registration Authority, RA) toimesta. Esimerkiksi pankkivarmenteen noudossa avainparin muodostus hoidetaan asiakkaan pankkiyhteysohjelmalla, hakijan rekisteröinti ja tunnistaminen hoidetaan pankin konttorissa Web Services -palvelusta sovittaessa ja

CA:n roolia hoitaa pankin tietojärjestelmiä hallinnoiva taho. PKI-järjestelmän on tarjottava ainakin seuraavat peruspalvelut:

1. Rekisteröinti – Varmennetta hakevan henkilön tietojen tarkastaminen.
2. Varmenteen luonti – Varmennepyynnön kirjoittaminen varmenteeksi ja tietojen allekirjoitus.
3. Varmenteen jakelu – Luodun varmenteen turvallinen jakelu haltijalle.
4. Varmennehakemiston ylläpito – Ajantasainen hakemisto kaikista CA:n myöntämistä varmenteista.
5. Sulkulistapalvelu – Varmenteiden hallintaan kuuluu osana sulkulista (Certificate Revocation List, CRL) Listalle lisätään kaikkien niiden varmenteiden sarjanumerot, jotka on jouduttu mitätöimään ennen niin vanhentumista. Reaaliaikaisia sulkulistakyselyitä varten on OSKP-protokolla. Myös varmenteen laajennetuissa tiedoissa on mukana URL-osoite tuoreinta sulkulistan noutoa varten.
6. Ylläpidolliset tukipalvelut – Myönnettyjen varmenteiden hakemiston ja sulkulistan ylläpidon lisäksi on joukko muita tukitoimenpiteitä kuten käyttäjien kyselyt, vikatilanteiden selvittelyt, varmenteiden uusinta niiden voimassaoloajan umpeutuessa ja niin edelleen.
7. Toimintakuvaus – PKI-järjestelmää ylläpitävän tahon tulee laatia kuvaustoiminnastaan, jota kutsutaan varmennekäytäntölausumaksi (Certificate Practice Statement, CPS). Lausuman tulisi olla Internet Engineering Task Forcen standardin RFC 2527 mukainen ja kattaa sisällöllisesti stadardin suosittelemat varmenteen luotettavuuteen ja tuottamiseen liittyvät asiat. [14; 15]

5.3 Samlink Customer CA

Samlink Customer CA toimii varmenteiden myöntäjänä Samlinkin Web Service Aineistosiirto palvelussa käytettävissä käyttäjävarmenteissa. Samlink Customer CA-järjestelmään liittyy yksi CA-varmenne, jossa käytetään seuraavan taulukon mukaisia tietoja.

Taulukko 4. Samlink Customer CA:n varmentajan varmenteen tiedot [15]

Kentän nimi	**Field name**	**Kentän sisältö**
Versio	Version	3
Sarjanumero	Serial number	80:8e:c2:f0:14:25:e2:a3:99:01:a5:12:06:71:19:39
Allekirjoitus-algoritmi	Signature algorithm	sha256WithRSAEncryption

Varmenteen myöntäjä	Issuer	C=FI, O=Samlink, CN=Samlink Customer CA
Voimassaolo-aika	Validity	Not Before: Aug 18 08:00:35 2009 GMT Not After : Aug 18 08:00:35 2034 GMT
Varmenteen haltija	Subject	C=FI, O=Samlink, CN=Samlink Customer CA
Varmenteen haltijan Julkisen avaimen tiedot	Subject public key info	Public Key Algorithm: rsaEncryption RSA Public Key: (4096 bit)
Varmentajan Julkisen avaimen tunniste	Authority key identifier	key-id:CA:80:38:33:93:8A:63:04:91:8D:05:69:56:68:42:35:E5:C7:FF:BC
Varmenteen haltijan Julkisen avaimen tunniste	Subject key Identifier	CA:80:38:33:93:8A:63:04:91:8D:05:69:56:68:42:35:E5:C7:FF:BC
Sulkulistan julkaisupaikka	CDP CRL distribution points	URL=ldap://194.252.124.241:389/cn=Samlink%20Customer%20CA,o=Samlink,c=fi?certificaterevocationlist;binary
Avaimen käyttötarkoituksen laajennus	Extended key usage	critical Digital Signature, Certificate Sign, CRL Sign

Samlink Customer CA:n myöntämät käyttäjävarmenteet ovat voimassa kaksi vuotta varmenteen allekirjoitushetkestä. Seuraavassa taulukossa on esitettynä WS-Aineistopalvelut käyttäjävarmenteessa käytetyt kentät:

Taulukko 5. Samlinkin Web Services -aineistosiitopalvelun käyttäjävarmenteissa käytetyt kentät [15.]

Kentän nimi	**Field name**	**Kentän sisältö**
Myöntäjä	Issuer	CN=Samlink Customer CA, O=Samlink, C=FI
Avaimen pituus	Key	2048 bits/RSA
Tiivisteet	Signature algorithm	SHA256
Yksikäsitteinen nimi	DN	SN,CN,O,C (Pankin sopimusjärjestelmästä tulleet arvot); muut pyynnön mukaan
Laajennukset		
Varmenteen haltijan Julkisen avaimen tunniste	Subject Key Identifier	Avaimen tiiviste 20-tavuisena

Varmentajan julkisen avaimen tunniste	Authority Key Identifier	KeyID=02 aa 0c 9e bd e9 48 81 27 08 28 e6 e8 de 14 f7 15 8c b9 b6
Revokointilistan julkaisuosoite	CDP CRL distribution points	[1] URL=ldap://194.252.124.241:389/cn=Samlink%20Customer%20CA,o=Samlink,c=fi?certificaterevocationlist;binary
Avaimen käyttötarkoitus	Key usage	Critical; Digital Signature, Non-Repudiation, Key Encipherment, Data Encipherment
CP – varmennpolitiikan tunnus		Arvona: Samlink Customer CA Varmenneperiaatteet WS-Aineistopalvelut varmenteita varten OID: 1.2.246.558.10.0970 4098.11.2 V.1.0

5.4 Varmennepalvelu – CertificateService

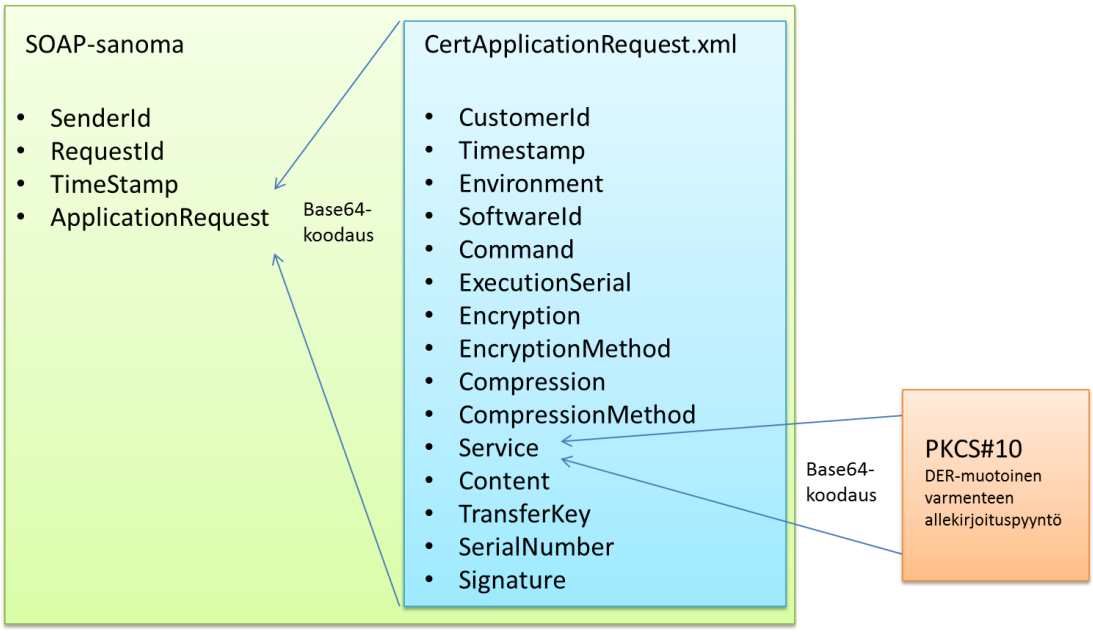
Aineistosiirroissa sanomien allekirjoitukseen käytettävien varmenteiden jakelussa hyödynnetään samoja Web Servicen tarjoamia teknisiä ratkaisuja kuin itse pankkiaineistojenkin siirrossa. SOAP-sanomien runko-osan rakennu muistuttaa pitkälti aineistosiirtopalvelusta tuttua rakennetta, mutta SOAP-sanomaa ei varmennepalvelussa allekirjoiteta sanoman otsikko-osioon. Samlinkin varmennepalvelu tukee vain yhtä SOAP-operaatiota, joka on nimeltään getCertificate

Aineistosiirtojen tapaan varsinainen käsiteltävä hyötykuormana on SOAP-bodyn ApplicationRequest-elementtiin sijoitettu Base64-koodattu XML-dokumentti. CertApplicationRequest-nimisen xml-dokumentin Content-elementiin sijoitetaan PKCS#10-standardin mukainen varmenteen allekirjoituspyyntö. Kun käyttäjä hakee ensimmäistä kertaa varmennetta WS-käyttäjätunnukselle, CertApplicationRequest-dokumentissa tunnistukseen käytetään TransferKey-nimistä elementtiä. Tämän elementin sisällöksi sanoman lähettäjä laittaa Web Services -sopimuksen yhteydessä kahdessa osassa toimitetun yhteensä 16 merkkisen kertakäyttöisen salasana. Kun aikaisemmin noudettua varmennetta uusitaan palvelun kautta TransferKey-elementti. korvataan XML-allekirjoituksella. Allekirjoitus tulee tehdä voimassaolevalla varmenteella. Tämän vuoksi on tärkeää muistaa lähettää varmenteen uusintapyyntö, ennen kuin käytössä oleva

varmenne vanhenee. Kaksi vuotta varmenteen allekirjoitushetkestä eteenpäin voimassa olevan varmenteen voi uusia, kun varmenteen voimassaolo aikaa on jäljellä alle 60 vuorokautta. Uuden varmenteen voi hakea vain, jos varmennetta ei ole vielä haettu tai nykyinen varmenne on alle 60 päivääavoimassa. Jos varmennetta yritetään uusia edellä mainittujen ehtojen vastaisesti, tuloksena palautetaan virheilmoitus seuraavan taulukon mukaisesti.

Taulukko 6. Varmenteen noudon ehtojen mukaiset vastaussanomat

avain	**käyttäjä**	**voimassaoloaika**	**kuvaus**
uusi	uusi	ei varmennetta	Normaali uuden luominen
uusi	vanha	alle60	Normaali uusinta
uusi	vanha	yli60	VIRHE:Varmennetta ei saa vielä uusia
vanha	uusi	alle60	VIRHE: Virheellinen käyttäjätieto
vanha	uusi	yli60	VIRHE: Virheellinen käyttäjätieto
vanha	vanha	alle60	VIRHE:Yksityinen avain pitää generoida uudestaan
vanha	vanha	yli60	Palautetaan vanha varmenne



Kuvio 27. Varmennepalvelun SOAP-sanoman muodostus ja elementit

Seuraavana on lueteltuna SOAP-sanoman elementit ja niiden käyttötarkoitus Samlinkin toteutuksen mukaan:

- SenderId - Viestin lähettäjän yksilöivä tunniste ja käyttäjätunnus. Arvo on sama kuin myöhemmin kuvattava CertApplicationRequestin CustomerId-elementissä ja varmennepyynnön Surname-kentässä (SN).

- RequestId - Lähetyksen yksilöivä tunniste, jonka tarkoitus on helpottaa ongelmien selvitetystyötä. Arvoa ei tarkisteta, onko samaa tunnistetta käytetty aikaisemmin.
- Timestamp - Aikaleima, joka kertoo milloin Application Request Header on luotu.
- ApplicationRequest - Sisältää Base64-koodattuna CertApplicationRequestin.

Seuraavan on listattuna Samlinkin toteutuksen mukaisen CertApplicationRequest-tiedosto rakenne, elementit ja niiden käyttötarkoitus:

- CustomerId - Varmenteen allekirjoituspyynnön tekijän yksilöivä tunniste ja käyttäjätunnus. Arvo on sama kuin aiemmin kuvatussa SenderId-elementissä ja varmennepyynnön Sur-name-kentässä (SN).
- Timestamp - Skeemassa pakollinen. Arvoa ei käytetä.
- Environment - Arvo: PRODUCTION Testiominaisuutta ei käytetä.
- SoftwareId - Ohjelmiston tarkka tunniste, jonka tarkoitus on helpottaa ongelmien selvitetystyötä.
- Command Arvo: GetCertificate
- ExecutionSerial - Ei käytetä
- Encryption - Ei käytetä
- EncryptionMethod - Ei käytetä
- Compression - Ei käytetä
- CompressionMethod - Ei käytetä
- Service - Skeemassa pakollinen. Käytetään oletusarvoa ISSUER.
- Content - Base64-koodattu varmenteen allekirjoituspyyntö (PKCS#10).
- TransferKey - Ensimmäisellä kerralla käytettävä kertakäyttöinen salasana (8+8). Kun asiakas on vastaanottanut allekirjoitetun varmenteen, käytetään varmennetta (Signature-elementti) asiakkaan tunnistamisessa, jolloin TransferKey-elementtiä ei saa enää esiintyä. Kertakäyttöinen salasana merkataan käytetyksi, kun allekirjoitettu varmenne lähtee asiakkaalle.
- SerialNumber - Ei käytetä
- Signature - Ensimmäisellä kerralla asiakas tunnistautuu käyttämällä kertakäyttöistä salasanaa (TransferKey-elementti). Sen jälkeen TransferKey-elementtiä ei enää käytetä ja vaan tunnistus tapahtuu XML-allekirjoituksen avulla.

Content-elementissä Base64-koodatun varmenteen allekirjoituspyynnön tulee olla X.509v3:n mukainen, DER-muotoinen, jossa varmenteen hakijan tiedot tulee olla subjectissa muodossa:

- Surname = Web Services -käyttäjätunnus
- CommonName = Pankin asiakastiedon mukainen nimi
- Organisation = Aineistopalvelut-???
- Country = FI

Organisaatio tiedon loppuosa vaihtelee pankin mukaan, mutta alkaa aina merkkijonolla Aineistopalvelut- .

Seuraavassa on listattuna CertApplicationResponce-dokumentin elementit ja niiden käyttötarkoitukset:

- CustomerId - Asiakkaan CertApplicationRequestissä käyttämä tunniste.
- Timestamp - Aikaleima, joka kertoo milloin CertApplicationResponse on luotu.
- ResponseCode - Katso taulukko 4.
- ResponseText - Katso taulukko 4.
- ExecutionSerial - Ei käytetä.
- Encrypted - Ei käytetä.
- EncryptionMethod - Ei käytetä.
- Compressed - Ei käytetä.
- CompressionMethod- -Ei käytetä
- CustomerExtension - Ei käytetä.
- Certificates - Sisältää yhden Certificate-elementin.
- Certificate - Sisältää Name-, Certificate- ja CertificateFormat-elementit.
- Name - Varmenteen subjekti esim.SURNAME=9923233, CN=YRITYS AB, O=Aineistopalvelut-Samlink, C=FI
- Certificate - Base64-koodattu allekirjoitettu varmenne (X509v3).
- CertificateFormat - Arvo: X509
- Signature - Samlink lisää XML-allekirjoituksen kaikille viesteille. Asiakkaan tulee varmistaa alle-kirjoituksen oikeellisuus.

Taulukko 7. CertApplicationResponsen vastauskoodit ja selitteet

ResponseCode	**ResponseText**
00	OK
05	Tuntematon sovelluspyyntö
06	Varmennetta ei saa vielä uusia
07	Virheellinen kätttäjätieto
08	Yksityinen avain pitää generoida uudelleen
12	Aineiston muodollinen tarkistus epäonninstui
26	Tekninen virhe
30	Tunnistus epäonnistui

6 Yhteenveto

Web Service -tekniikka tuo mukanaan asiakasohjelmistojen kehittäjille mahdollisuuden juostavampaa ja nopeampaan ohjelmistokehitykseen nykypäiväsillä välineillä. Uusi tekniikka luo myös pankin palveluihin huomattavaa kehitys potentiaalia, kun sanomien laajennusmahdollisuudet ovat erittäin mittavat, uusien ominaisuuksien standardointi jatkuvaa maailman laajuisen ja monimuotoisen käytön ansiosta.

Pankkiyhteyskäyttöön määritelty Web Services -kanava on kieltämättä monitasoisen sanomarakenteen useine allekirjoituksineen ja datalle tehtävien Base64-koodauksien vuoksi, aika monimutkainen. Hyötydatalle tehtävät Base64-koodaukset myös kasvattavat sanomien kokoa aika merkittävästi. Vaikka lähetettävän datan määrä on noussut Web Services-kanavan ja XML-pohjaisten aineistojen johdosta, on melko harvinaista kuitenkaan hyödyntää aineistojen pakkausta. Kotimaisen suoraveloituksen päättyminen tammikuussa 2014 johti Finvoice-verkkolaskutuksen räjähdysmäiseen kasvuun, sen myötä välitettävien aineistojen koko on viime aikoina luonut ainakin suurimpien toimijoiden keskuudessa tarvetta aineistojen pakkaukselle.

Vanhaa PATU-linjaa tukevilla pankeilla on selkeästi kova tahtotila päästä luopumaan vanhasta yhteyskäytännöstä kokonaan. Melko kunnioitettavaan n. 25 vuoden ikään ehtinyt PATU-yhteyskäytäntö alkaa nykymittapuulla olemaan jo yksi tietotekniikan dinosauruksista ja selväähän tuo lienee, että kahden rinnakkaisen yhteyskanavan ylläpito on kustannus sekä kehityksen jarru. Suurimmat pankkiyhteysohjelmistoja tarjoavat ohjelmistotalot ovat tukeneet tuotteissaan Web Services-kanavan käyttöä jo useamman vuoden, mutta liiketoiminta sektorilla vallitseva ilmapiiri PATU:n lopettamisesta saanut nyt viime viimeisen vuoden aikana pienimmät ohjelmatoimittajat kehittämään tuotteitaan. Tämä on näkynyt selkeänä kasvuna haastavampien kehitystyöhön liittyvien palvelupyyntöjen ja sekä testaustuen kasvaneena kysyntänä.

Opinnäytetyötä voidaan pitää onnistuneena sille asetettujen tavoitteiden valossa. Ohjelmistotalojen palvelupyyntöjen ratkaisu on nopeutunut huomattavasti kohentuneen osaamisen myötä, vaikka määrällisesti kasvu on ollut tuntuvaa. Ongelmatilanteiden selvittelyä varten on otettu käyttöön uusia työkaluja, joiden avulla sanomien ja xml-dokumenttien tarkastus prosessit on tehostunut. Uudistettu ja päivitetty dokumentaatio ohjelmistotalojen käyttöön julkaistaan lähiviikkojen aikana, jonka tulisi vaikuttaa palvelupyyntöjen määrään laskevasti. Työn yhtenä tavoitteena olisi saada työkalujen ja ohjeistuksen avulla ongelmatilanteiden selvitystä helpotettua niin paljon, että tekniikkaan vähemmän tutustuneen olisi helpompi perehdyttää ratkaisemaan ainakin tyypillisimpiä ohjelmistokehittäjien kohtaamia ongelmia. Tämänkin tilanteen näen parantuneen. Uuden lokityökalun ja parantuneiden virheilmoitusten myötä ongelman aiheuttaja on helpompi löytää ja ymmärtää.

Lähteet

1 Samlink, Pankkiyhteys käyttäjänohje. 2011. Verkkodokumentti. Viitattu 15.02.2014. http://193.65.156.50/pdf/Pankkiyhteysk.pdf.

2 Nordea, OP-Pohjola Group, Sampo Bank. 2008. Security and Message Specification for Financial Messages using Web Services. Verkkodokumentti. Viitattu 15.02.2014.http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/WebServices_Messages_20081022_105.pdf.

3 W3School , Introduction to Web services, Verkkodokumentti. Viitattu 15.02.2014 http://www.w3schools.com/webservices/ws_intro.asp.

4 Tuikka, Kanala, 2001. XML-Ohjelmoinnin perusteet , Edita: Helsinki.

5 W3School, XML tutorial, Verkkodokumentti. Viitattu 17.02.2014. http://www.w3schools.com/xml/default.asp.

6 Mediakasvatus- ja kuvaohjelmakeskus, Elokuvien ikärajat, Viitattu 10.3.2014 http://www.meku.fi/index.php?option=com_content&view=article&id=23&Itemid=390&lang=fi.

7 Newcomer, Eric. Understandin Web Services : XML, WSDL, SOAP and UDDI. 2002. Pearson: Indianapolis.

8 W3C, Web Services Description Language WSDL 1.1. 2001. Viitattu 13.3.2014 http://www.w3.org/TR/wsdl.

9 Jävinen. 2002. Hajautetut verkkopalvelut. Docendo Finland Oy, SanomaWSOY-konserni.

10 W3C, Simple Object Access Protocol (SOAP). Viitattu 15.3.2014 http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

11 W3C, XML Signature Syntax and Processing. Viitattu 21.4.2014 http://www.w3.org/TR/xmldsig-core/.

12 OASIS. Web Services Security. 2005. Viitattu 22.4.2014 https://www.oasis-open.org/committees/download.php/13383/wss-v1.1-spec-pr-x509TokenProfile-01.htm.

13 Samlinkin tekninen rajapintakuvaus ohjelmistotaloille 2010. Verkkodokumentti. Viitattu 24.4.2014 http://193.65.156.50/pdf/PalvelukuvausWS.pdf.

14 Järvinen. 2003. Salausmenetelmät, Decondo Finland Oy, Sanoma WSOY-konserni.

15 Samlink, SAMLINK CUSTOMER CA Varmennuskäytäntö 2009, Verkkodokumentti. Viitattu 27.4.2014 http://193.65.156.50/pdf/CAVarmennuskaytanto.pdf.

Varmennepalvelun WSDL-dokumnetti

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://mlp.op.fi/OPCertificateService"
xmlns:tns="http://mlp.op.fi/OPCertificateService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
<xsd:schema targetNamespace="http://mlp.op.fi/OPCertificateService" elementFormDefault="qualified"
attributeFormDefault="qualified">
<xsd:complexType name="CertificateRequestHeader">
<xsd:sequence>
<xsd:element name="SenderId" type="xsd:string" nillable="false"/>
<xsd:element name="RequestId" type="xsd:string" nillable="false"/>
<xsd:element name="Timestamp" type="xsd:dateTime" nillable="false"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CertificateResponseHeader">
<xsd:sequence>
<xsd:element name="SenderId" type="xsd:string" nillable="false"/>
<xsd:element name="RequestId" type="xsd:string" nillable="false"/>
<xsd:element name="Timestamp" type="xsd:dateTime" nillable="false"/>
<xsd:element name="ResponseCode" type="xsd:string" nillable="true"/>
<xsd:element name="ResponseText" type="xsd:string" nillable="true"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetCertificateRequest">
<xsd:sequence>
<xsd:element name="RequestHeader" type="tns:CertificateRequestHeader" nillable="false"/>
<xsd:element name="ApplicationRequest" type="xsd:base64Binary" nillable="false"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetCertificateResponse">
<xsd:sequence>
<xsd:element name="ResponseHeader" type="tns:CertificateResponseHeader" nillable="false"/>
<xsd:element name="ApplicationResponse" type="xsd:base64Binary" nillable="false"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="CertificateServiceFaultDetail">
<xsd:sequence>
<xsd:element minOccurs="0" maxOccurs="1" name="category" type="xsd:string"/>
<xsd:element minOccurs="0" maxOccurs="1" name="code" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
<xsd:schema targetNamespace="http://mlp.op.fi/OPCertificateService" elementFormDefault="qualified"
attributeFormDefault="qualified">
<xsd:element name="getCertificatein" type="tns:GetCertificateRequest"/>
<xsd:element name="getCertificateout" type="tns:GetCertificateResponse"/>
<xsd:element name="certificateServiceFaultElement" type="tns:CertificateServiceFaultDetail"/>
</xsd:schema>
</wsdl:types>
<wsdl:message name="certificateServiceFault">
<wsdl:part name="certificateServiceFault" element="tns:certificateServiceFaultElement"/>
</wsdl:message>
<wsdl:message name="getCertificateRequest">
<wsdl:part element="tns:getCertificatein" name="getCertificatein"/>
</wsdl:message>
<wsdl:message name="getCertificateResponse">
<wsdl:part element="tns:getCertificateout" name="getCertificateout"/>
</wsdl:message>
<wsdl:portType name="OPCertificateServicePortType">
<wsdl:operation name="getCertificate">
<wsdl:input message="tns:getCertificateRequest" name="getCertificateRequest"/>
<wsdl:output message="tns:getCertificateResponse" name="getCertificateResponse"/>
<wsdl:fault message="tns:certificateServiceFault" name="certificateServiceFault"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="OPCertificateServiceHttpBinding" type="tns:OPCertificateServicePortType">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getCertificate">
<soap:operation soapAction=""/>
<wsdl:input name="getCertificateRequest">
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output name="getCertificateResponse">
```

```
<soap:body use="literal"/>
</wsdl:output>
<wsdl:fault name="certificateServiceFault">
<soap:fault use="literal"/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="OPCertificateService">
<wsdl:port binding="tns:OPCertificateServiceHttpBinding" name="OPCertificateServiceHttpPort">
<soap:address location="http://domain.fi:1001/wsdl/CertificateService.xml"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```