Opinnäytetyö (AMK)

Tietotekniikka

Hyvinvointiteknologia

2015

Sami Suo-Heikki

VIITESIIRTOAINEISTON NOUTO OSUUSPANKISTA KÄYTTÄEN WEB SERVICES -YHTEYSKÄYTÄNTÖÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Hyvinvointiteknologia

2015 | 42 sivua

Tiina Ferm

Sami Suo-Heikki

VIITESIIRTOAINEISTON NOUTO OSUUSPANKISTA KÄYTTÄEN WEB SERVICES -YHTEYSKÄYTÄNTÖÄ

Tässä opinnäytetyössä oli tavoitteena toteuttaa Tehden-ohjelmistoon viitesiirtoaineiston nouto Osuuspankista käyttäen Web Services -yhteyskäytäntöä. Tehden-ohjelmisto on verkkopohjainen toiminannanohjausjärjestelmä, joka auttaa yrittäjiä jokapäiväisessä työssään. Työssä käsiteltiin yhteyskäytännön soveltamista ja käyttöönottoa olemassa olevaan ohjelmistoon. Tämän lisäksi esiteltiin yhteyskäytännössä käytettäviä eri tekniikoita, yhteyskäytännön mahdollisuuksia ja sen tuomaa muutosta pankkiyhteyksiin.

Web Services eli yrityksen pankkiyhteys -kanava on tarkoitettu konekielisen pankkiaineiston siirtoon yrityksen ja pankin välillä. Kanavaa voidaan käyttää maksuaineiston lähettämiseen ja noutamiseen pankista. Web Services -kanava perustuu nykyaikaisiin XML-sanomiin ja PKI-teknologioihin ja seuraa yhtenäisen euromaksualueen standardeja.

Toteutettava kokonaisuus koostui PHP-ohjelmointikielellä toteutettavasta PKI-varmenteiden hallinnasta, pyyntösanomien muodostamisesta ja allekirjoittamisesta, SOAP-rajapinnasta sekä vastaanotettujen aineistoiden käsittelystä. Maksuaineistot tallennetaan PostgreSQL-relaatiotietokantaan. Valmis kokonaisuus sisältää yrityksen tunnusten varmentamisen ja viitesiirtoaineiston ajastetun sekä manuaaliseen noudon Osuuspankin Web Services -kanavasta.

Opinnäytetyön tuloksena syntynyt kokonaisuus on opinnäytetyön kirjoitushetkellä jo käytössä usealla yrityksellä. Toteutettu kokonaisuus loi lisäksi lähes valmiin pohjan muiden pankkien yhteyskäytännön toteuttamiseen Tehden-ohjelmistossa.

ASIASANAT:

Web Services, SOAP, XML, SEPA, PKI, PHP, digitaalinen allekirjoitus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Health Informatics

2015 | 42 pages

Tiina Ferm

Sami Suo-Heikki

FETCHING PAYMENT INFORMATION FROM OSUUSPANKKI USING WEB SERVICES

The aim of the present Bachelor's thesis is to implement a payment information fetch from Osuuspankki, a Finnish bank, to the Tehden system using the Web Services -protocol. The Tehden system is a web-based ERP system that facilitates the flow of information for business in the fields of wellbeing and commerce. The thesis describes the application and deployment of the protocol into an existing software. Moreover, it also covers the different technologies used in the protocol, the protocol's possibilities and its effect on banking connections.

Web Services are designed for the transfer of electronic banking material between a company and a bank and it is used for sending and retrieving payment information from the bank. A Web Services channel is based on modern XML messages and PKI technology and it follows the current standards of the Single Euro Payments Area.

The implementation of the Web Services protocol included a module written with PHP that contains a PKI certificate management, the formation and signing of request messages, the SOAP API, as well as processing the received data. The payment information is saved in a PostgreSQL relational database. Ready implementation is used for verifying company's user information and fetching payment information both manually and automatically from the Osuuspankki Web Services channel.

The result of the thesis, the implementation of payment information fetch, is at the time of writing already used by several companies. The implementation works as a foundation for implementing Web Services protocol to others banks in the Tehden system.

KEYWORDS:

Web Services, SOAP, XML, SEPA, PKI, PHP, digital signature

SISÄLTÖ

LIITTEET

KOODIT

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET JA SANASTO

B2B	Business to business.
B2C	Business to consumer.
Base64	Algoritmi binäärisen tiedon koodittamiseen ja purkamiseen.
C2B	Customer to bank.
DER	PKI-sertifikaatin muoto.
GZIP	Pakkausohjelma.
FTP	FTP: File Transfer Protocol, tiedon-siirtomenetelmä.
HTTPS	Suojattu tiedonsiirtoprotokolla.
OpenSSL	Avoimen lähdekoodin kirjasto, mikä sisältää kryptograafisia funktioita.
PATU	Pankkien tiedonsiirtoyhteyksien tietoturva.
PEM	PKI-sertifikaatin muoto.
PHP	PHP: Hypertext Preprocessor, ohjelmointi-kieli.
RFC1952	Pakkausalgoritmi.
RSA	Julkisen avaimen salausalgoritmi.
SHA-1	Tiivistefunktio.
Tehden-ohjelmisto	Web-toiminnanohjausjärjestelmä.
WS	Web Services.
WS-kanava	Web Services kanava.

1 JOHDANTO

Yhtenäisen euromaksualueen eli SEPAn muodostumisen myötä Euroopan alueen pankkien tiedonsiirtomenetelmä on pyritty korvaamaan kehittyneemmällä, turvallisemmalla ja kansainvälisiin standardeihin perustuvalla palvelulla. Suomessa vanha PATU-tietoturvaratkaisu ja FTP-tiedostonsiirto on korvattu Web Services -palvelulla ja PKI-standardeilla. (Finanssialan keskusliitto 2010.)

Web Services -yhteyskäytäntö on Suomen pankkien ja pankkien yritysasiakkaiden välinen tiedonsiirtoprotokolla. Yhteyskäytäntöä käytetään asiakkaan ja pankin välisiin maksuliikeaineistoiden välitykseen. Web Services -yhteyskäytännön avulla yritykset voivat esimerkiksi noutaa sekä lähettää pankkiin erilaisia maksuliikeaineistoja, kuten e-laskuaineistoja ja viitesiirtoaineistoja. (Finanssialan keskusliitto 2010.)

Uusi yhteyskäytäntö on XML-pohjainen kansainvälisiin standardeihin perustuva turvallinen tiedonsiirtomenetelmä (Finanssialan keskusliitto 2013).

Tämän opinnäytetyön tavoitteena on luoda toimiva yhteys Tehden-toiminnanohjausjärjestelmän sekä Osuuspankin Web Services -kanavan välille. WS-kanavaa tullaan käyttämään viitesiirtoaineiston ajastettuun eli automaattiseen noutoon. Uusi toteutus korvaa aiemman ohjelmistossa olevan menetelmän. Aiemmassa toteutuksessa käyttäjä joutuu itse noutamaan uudet viitesiirtoaineistot verkkopankistaan ja syöttämään ne manuaalisesti Tehden-ohjelmistoon.

WS-kanavan käyttö mahdollistaa viitesiirtoaineiston automatisoidun sekä manuaalisen noudon Tehden-ohjelmistossa. Käyttäen WS-kanavaa aineiston nouto on asiakkaalle myös halvempaa. Opinnäytetyöhön valittiin Osuuspankki muiden pankkien sijasta, koska merkittävä osa Tehden Oy:n käyttäjistä ovat Osuuspankin yritysasiakkaita. Tehden Oy myös itse on Osuuspankin asiakas, mikä helpottaa ja nopeuttaa Web Services -yhteyskäytännön testaamista sekä käyttöönottoa.

Opinnäytetyön valmis toteutus sisältää WS-kanavan käyttöön vaadittavat teknilliset ominaisuudet. Eri ominaisuudet ovat PKI-avainparien luonti sekä niiden varmentaminen, lähetyksien digitaalinen allekirjoittaminen, viitesiirtoaineiston nouto, aineiston tallentaminen tietokantaan ja noudetun aineiston visualisointi käyttäjälle.

Työ toteutetaan pääosin käyttäen PHP-ohjelmointikieltä, jolla luodaan tarvittavat luokkakirjastot sekä Web Services -rajapinta. Rajapinnan yli kulkeva tieto on XML-pohjaista, jonka käsittely myös toteutetaan PHP:lla. Web Services -sopimukseen liittyvät tiedot, kuten käyttäjätunnukset ja sertifikaatit tallennetaan sekä noudetaan PostgreSQL-tietokannasta. Käyttöliittymän luonnissa käytetään HTML-, CSS- ja JavaScript-ohjelmointikieliä.

Web Services -yhteyskäytännössä lähes kaikki käsiteltävä aineisto on ISO-2002 -standardiin perustuvaa XML-tietoa. Tämän vuoksi opinnäytetyössä ensin käsitellään XML-merkintäkieltä ja sen eri ominaisuuksia. Luvussa 3 käsitellään Web Services -yhteyskäytännön ominaisuuksia ja sen eri mahdollisuuksia. Luvussa 4 käsitellään Tehden-ohjelmistoon toteutettua WS-rajapintaa sekä sen käyttöä eri tilanteissa.

Opinnäytetyötä tehtiin vakituisessa työsuhteessa Tehden Oy:ssa. Tehden on vuonna 2007 perustettu suomalainen ohjelmistotalo, joka tarjoaa liiketoimintaa tehostavia yritysohjelmistoja. Yrityksen toiminta alkoi nimellä e-ngine Oy, mutta Tehden-ohjelmiston lanseerauksesta lähtien yrityksen virallinen nimi on ollut Tehden Oy. (Tehden Oy 2015.)

Tehden-toiminnanohjausjärjestelmä on kotimainen pilvipalveluohjelmisto, joka auttaa yrittäjiä jokapäiväisessä työssään. Toiminnanohjausjärjestelmä sopii joustavuuden vuoksi isoille ketjuille ja toiminimillekin. Tehden-ohjelmisto sisältää kaupankäyntiin tärkeitä osia, kuten kassa, ajanvaraus, laskutus, tuote- ja varastohallinta, asiakashallinta, osto- ja myyntitilausjärjestelmän sekä verkkokaupan. Tehden-ohjelmistoa käyttää lähes 500 yrittäjää ympäri Suomea. (Tehden Oy 2015.)

2 XML-MERKINTÄKIELI

XML on rakenteellinen merkintäkieli tarkoitettu rakenteellisen tiedon esittämiseen. XML:llä voidaan auttaa jäsentämään laajojakin tietomääriä selkeämmin. Rakenteellinen tieto voi sisältää muun muassa osoitetietoja, sähköisen kaupankäynnin viestejä tai varastosaldoja. XML ei ole varsinainen ohjelmointikieli, vaan muita ohjelmointikieliä käytetään sen käsittelyssä. Lähes jokainen ohjelmointikieli tukee XML:ää. (W3C 2014.)

XML muistuttaa paljon HTML-kieltä, jolla luodaan WWW-sivuja. Sekä XML:ssä että HTML:ssä käytetään tageja sekä attribuutteja. Toisin kuin HTML:ssä, XML-tiedon rakennetta ei ole ennalta määrätty, vaan rakenteen tulkinnan tekee täysin tietoja käsittelevä ohjelmisto tai henkilö. Esimerkiksi HTML-kielessä koodi ”<title>” tarkoittaa dokumentin otsikkoa, mutta XML:ssä sille voi antaa minkä tahansa merkityksen. (O’Reilly Media, Inc. 2014.)

XML-tiedosto on hierarkkinen dokumenttipuu, joka rakentuu prologista, juuri- ja lapsielementeistä sekä niiden attribuuteista. Juurielementtejä on XML-tiedostossa vain yksi ja sen sisäpuolelle rakentuvat lapsielementit. Lapsielementit voivat myös sisältää omia lapsielementtejä. (W3C 2014.)

2.1 XML-dokumentti

XML-dokumentti alkaa ensimmäisellä rivillä olevalla prologilla. Prologi sisältää tiedon dokumentin eri ominaisuuksista, kuten XML-version ja käytössä olevan merkistökoodauksen. Prologi voi myös sisältää tiedon voidaanko myöhemmin viitattu DOCTYPE jättää lukematta. Prologi ei ole pakollinen vaan dokumentin voi myös aloittaa juurielementistä. Jos prologia käytetään sitä ei saa edeltää mikään muu merkki. Koodissa 1 XML-dokumentille asetetaan XML-versio 1.0 sekä utf-8 merkistökoodaus. (W3C 2010.)

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Tilitapahtumat>
    <Tilitapahtuma id="1546">
        <Kirjauspäivä>08.07.2015</Kirjauspäivä>
        <Saaja>K-market</Saaja>
        <Maksaja/>
        <Tilinumero>FI74 1234 5678 9012 03</Tilinumero>
        <Tapahtumatyyppi>PKORTTIMAKSU</Tapahtumatyyppi>
        <Määrä yksikkö="EURO">-16</Määrä>
    </Tilitapahtuma>
    <Tilitapahtuma id="1545">
        <Kirjauspäivä>03.07.2015</Kirjauspäivä>
        <Saaja>S-market</Saaja>
        <Maksaja/>
        <Tilinumero>FI74 1234 5678 9012 02</Tilinumero>
        <Tapahtumatyyppi>PKORTTIMAKSU</Tapahtumatyyppi>
        <Määrä yksikkö="EURO">-42</Määrä>
    </Tilitapahtuma>
    <Tilitapahtuma id="1544">
        <Kirjauspäivä>01.07.2015</Kirjauspäivä>
        <Saaja/>
        <Maksaja>Työnantaja</Maksaja>
        <Tilinumero>FI74 1234 5678 9012 01</Tilinumero>
        <Tapahtumatyyppi>PALKKA</Tapahtumatyyppi>
        <Määrä yksikkö="EURO">2000</Määrä>
    </Tilitapahtuma>
</Tilitapahtumat>
```

Koodi 1. XML-dokumentin sisältö.

XML-dokumentin alkaessa prologilla toisella rivillä on juurielementti. XML-dokumentissa voi olla vain yksi juurielementti, jonka alle muodostetaan lapsielementeillä dokumentin rakenne ja sisältö. Koodissa 1 Tilitapahtumat on juurielementti, joka sisältää kolme Tilitapahtuma-elementtiä jolla jokaisella on 6 tilitapahtumaa kuvaavaa lapsielementtiä.

Kuten HTML-kielessä, XML-elementeille on mahdollista antaa attribuutteja. Attribuuteilla kuvataan tietoja, jolla voi tarkentaa elementin sisältämää tietoa. Esimerkiksi koodissa 1 tilitapahtumien määrä-elementille annetaan yksikkö ”EURO”, joka tarkentaa määrän sisältöä. Tällä viitataan tilitapahtuman valuutan olevan Euro ja näin valuutalle ei tarvitse määrittää omaa elementtiä. Attribuutin tulee aina olla lainausmerkkien sisällä (W3Schools 2015). XML-elementeillä ei tarvitse olla arvoa, vaan ne voivat olla tyhjiä. Näissä tapauksissa elementin voi merkitä ilman

lopputagia ja lisäämällä kauttaviivan ensimmäisen tagin loppuun. Koodissa 1 ensimmäisessä tilitapahtumassa Maksaja-elementti on ilman sisältöä, koska kyseessä ei ole pano vaan on pankkikorttimaksu.

2.2 XML-nimiavaruudet

XML-nimiavaruudet tarjoavat mahdollisuuden nimetä elementtejä nimiavaruuksien mukaan. Nimiavaruuksien hyöty on, että samaa elementin nimeä voidaan käyttää kahdessa tai useammassakin eri merkityksessä. (Microsoft 2002.)

Koodissa 2 on XML-dokumentti, joka sisältää kaksi table-elementtiä. Ensimmäisessä table tarkoittaa HTML-kielen taulukkoa, joka sisältää yhden rivin (tr) ja kaksi kolumnia (td). Jälkimmäisellä table-elementillä tarkoitetaan pöytää, jolle määritellään omat elementit. Koodissa 2 esitetty XML-dokumentti kuitenkin aiheuttaa konfliktin kahden table-elementin välille, koska käsittelevä sovellus ei tiedä miten nämä elementit tulee käsitellä. Samannimiset elementit vaativat samat lapsielementit. (Microsoft 2002.)

```
1  <root>
2      <table>
3          <tr>
4              <td>Apples</td>
5              <td>Bananas</td>
6          </tr>
7      </table>
8      <table>
9          <name>African Coffee Table</name>
10         <width>80</width>
11         <length>120</length>
12     </table>
13 </root>
```

Koodi 2. Ristiriitainen XML-dokumentti.

Ratkaisuna nimeämisongelmaan on käyttää nimiavaruuksia kuvaamaan tarkemmin eri elementtejä. Nimiavaruus tulee määritellä xmlns-attribuutilla. Määrittelyn voi tehdä mille tahansa elementille. Yleinen tapa on luoda nimiavaruus-määri-

telmä jo juurielementissä, jolloin sitä voidaan käyttää koko dokumentissa. Nimiavaruudet koskevat aina määrittelyn elementtiä sekä kyseisen elementin lapsielementtejä. Koodissa 3 on lisätty juurielementtiin kaksi nimiavaruutta, joita käyttämällä on erotettu kaksi table-elementtiä aiheuttamasta nimeämisongelmaa. (Microsoft 2002.)

```
<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>
```

Koodi 3. Ratkaisu ristiriitaiseen XML-dokumenttiin.

Nimiavaruuksille tulee antaa arvoksi uniikki nimi (Microsoft 2002). Yleinen tapa, kuten koodissa 3, on antaa arvoksi verkko-osoite, joka sisältää tietoa käytetystä nimiavaruudesta.

2.3 XML-skeema

XML-skeema kuvaa ja rajoittaa XML-dokumentin rakennetta. XML-skeemaa käyttäessä rakenne määritellään XML-mallitiedostossa, jonka mukainen XML-dokumentin tulee olla. Skeemalla on mahdollista luoda yhteinen sanasto XML:n rakentamiseen, joita myös tietokoneet ymmärtävät. (W3C 2004.)

XML-skeemalla on monta käyttötarkoitusta kuten elementtien tietotyypittäminen ja rajoituksien luonti XML-dokumentteihin. Elementtien tietotyypittämisellä on mahdollista rajoittaa minkälaista tietoa elementti saa sisältää ja rajoituksilla voi esimerkiksi määrittää käytössä olevat lapsielementit. Tietotyypit jaetaan yksin-

kertaisiin ja monimuotoisiin. Yksinkertaisiin tietotyyppeihin lukeutuu muun muassa merkkijono (string), totuusarvo (boolean) sekä päivämäärä (date). Monimuotoisiin tietotyypittämisiin lukeutuu tilanteet, jossa elementille annetaan useampi kuin yksi rajoite. Koodissa 4 name-elementille on asetettu tyypiksi string-arvo, joten se saa sisältää vain merkkijonon. (W3C 2004.)

Yksinkertaisen tietotyypin lisäksi elementille voidaan antaa useampi rajoite tai ehto. Koodissa 4 Country-elementille on asetettu continent-attribuutti pakolliseksi ja sen tyypiksi merkkijono.

```
<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="country" type="Country">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="continent" type="xs:string" use="required" />
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="Country">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="population" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Koodi 4. XML-skeema tietotyyppejä.

3 WEB SERVICES -YHTEYSKÄYTÄNTÖ

Yhtenäisen euromaksualueen eli SEPAn muodostumisen myötä Suomessa yritykset ja pankit siirtyvät XML-pohjaiseen SEPA-maksuaineistoon Web Services -yhteyskäytännön avulla. Suomi on ottanut uuden maksupalvelulain voimaan vuonna 2009 ja siirtymistä on toteutettu sen jälkeen. (Luoto 2008.)

Web Services -yhteyskäytäntö on nykyaikainen ja turvallinen tietoliikenneratkaisu yrityksen taloushallinnon ja pankin järjestelmien välisiin yhteyksiin välittämään maksuliikeaineistoja. Yksinkertaistettuna Web Services tarkoittaa WWW-pohjaisia ohjelmointirajapintoja. Etuina uudessa yhteyskäytännössä on, että yhteyden voi automatisoida ja ajastaa sekä tietoliikenneyhteys on aina salattu, eikä erillistä VPN-yhteyttä tarvita. (Nordea 2015.)

3.1 SEPA – Yhtenäinen euromaksualue

SEPA eli yhtenäinen euromaksualue on lyhenne englanninkielisestä termistä Single Payments Area. Euroopan Unionin SEPA-hankkeen tavoitteena on saavuttaa mahdollisimman automaattinen maksujen käsittely yhteisin standardein, sopia yhteisistä toimintatavoista, tehostaa sekä kotimaan- että ulkomaanmaksujen käsittelyä sekä huolehtia kustannustehokkuudesta Euroopassa. Hanke kattaa kaikki käteistä lukuun ottamatta tärkeimmät maksutavat, kuten tilisiirrot, korttimaksut sekä suoraveloituksen. Pankit voivat tarjota asiakkailleen muitakin palveluita järjestelmän puitteissa. (Suomen Pankki 2015.)

Sepa-tilisiirrot on otettu käyttöön tammikuussa 2008 ja nyt yli 4 000 pankkia maailmassa on sitoutunut toimimaan SEPAn säännösten mukaisesti. Vastuu säännösten vaatimista muutoksista on kuitenkin lähinnä pankkialalla itsellään. Suomessa muutos uuteen järjestelmään on toteutettu Web Services -palvelulla, jonka käyttöönotosta ja kuvauksista ovat sopineet yhdessä Nordea, OP-Pohjola-ryhmä sekä Sampo Pankki. Edellä mainitut pankit vastaavat palvelussa vaadittavista sanomakuvauksista, sekä ApplicationRequest XML - että ApplicationResponse XML-skeemojen yhteisestä dokumentoinnista. Yhteisten turvallisuus- ja

sanomamäärityksien lisäksi kukin pankki julkaisee oman pankkikohtaisen dokumentin, joka sisältää yksityiskohtaisemmat tiedot pankin WS-kanavan käytöstä. (Finanssialan Keskusliitto 2014.)

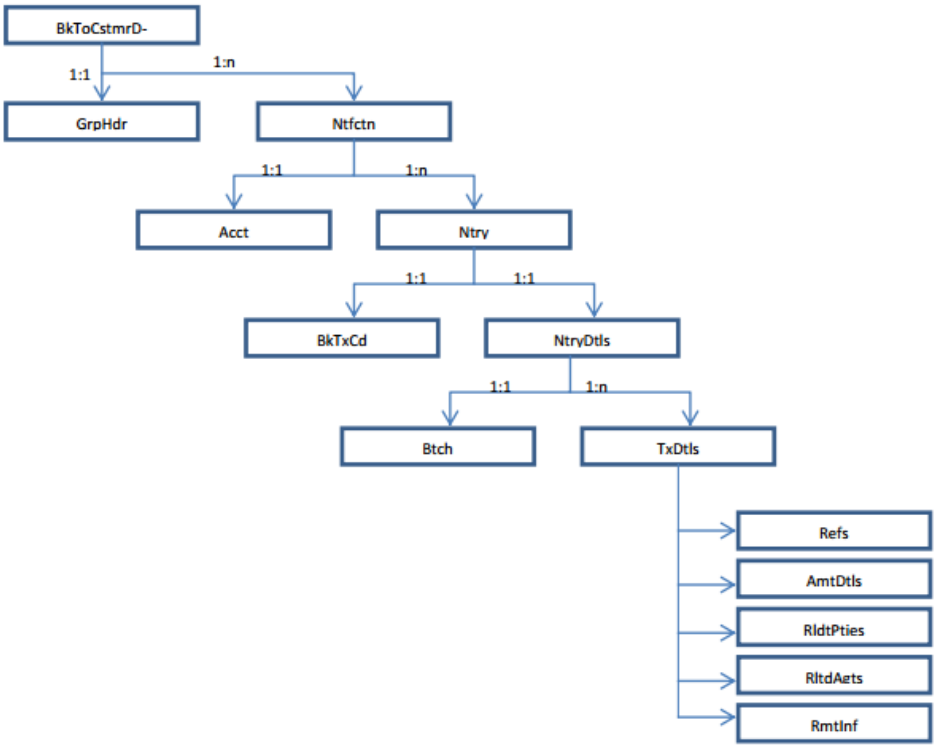
3.1.1 ISO 20022 -standardi

Suomen Web Services -palvelu sekä SEPA perustuu ISO 20022 -standardiin, joka on määritys XML-pohjaisista maksuaineistoista. ISO 20022 on ISO-organisaation määrittelemä sanomavälitysjärjestelmä, jonka sanomat on tarkoitettu yritysten ja pankkien väliseen viestintään. (ISO 2015 s 46.)

ISO 20022 -sanomia on useita eri luokkia. Esimerkkinä camt.054.001.02-luokka on XML-standardi, mikä kuvaa tapaa muodostaa pankkitilin viitteellisiä tapahtumia. Tämä standardi ja luokka ovat käytössä viitesiirtoaineistoa noudettaessa. Kuvassa 1 on esitetty camt.054.001.02 luokan XML-rakenne sekä sen XML-dokumentissa esiintyvät elementit. Kuvassa esitetty sarakkeiden toistuvuus on seuraava:

- 1:1 tieto on käytössä aina
- 0:1 tieto voi olla käytössä tai puuttua
- 1:n tieto on vähintään yhden kerran, mutta voi olla useamminkin

(OP-Palvelut Oy 2014.)



Kuva 1. camt.054.001.02-luokan XML-rakenne (Samlink 2013).

3.1.2 SEPAn vaikutus pankkiyhteyksiin

Suomen liittyessä SEPAan vanha PATU-tietoturvaratkaisu ja FTP-tiedostonsiirto ovat korvattu Web Service -palvelulla ja PKI-standardeilla. Web Services -palvelu täyttää SEPAn asettamat standardit sekä vaatimukset ja suurin osa Suomen pankeista nykyisin tukee Web Services -yhteyttä. Myös usea pankki on jo poistunut vanhasta eräsiirto/PATU ratkaisuista ja tarjoaa yrityksille ainoastaan Web Services -palvelua. OP-pohjola tukee vielä toistaiseksi Web Services -yhteyskäytännön rinnalla myös FTP ja PATU -ratkaisuja. (Finanssialan Keskusliitto 2015.)

3.2 Web Services -yhteyskäytännön käyttö

Web Services -palvelu tarjoaa yrityksille mahdollisuuden hoitaa yrityksen maksuliikenteen suoraan omasta taloushallinnon ohjelmistosta. WS-kanavalla yritys voi

lähettää sekä noutaa pankista maksuliikeaineistoja kuten tiliotteita, C2B-maksuaineistoja, e-laskuaineistoja ja niiden ilmoitussanomia. Aineistojen noudossa yrityksen ohjelmisto lähettää palvelupyynnön ja saa WS-kanavasta välittömästi vastauksen. Aineiston lähetyksessä pankki kuittaa vastaanottaneensa lähetyksen, minkä jälkeen se jää pankkiin odottamaan käsittelyä. Yrityksen tekemät palvelupyynnöt sisältävät allekirjoitetun SOAP-kirjekuoren, joka sisältää pankkiin lähetettävän aineiston tai pyydetyn palvelupyynnön. (OP-Pohjola-ryhmä 2011.)

Ennen palvelun käyttöönottoa yrityksen tulee tehdä kirjallinen sopimus Osuuspankin tai muun pankin kanssa. Sopimuksen luonnin yhteydessä yritys saa WS-kanavaan käyttäjätunnuksen sekä siirtoavaimen ensimmäisen osan. Jälkimmäinen osa siirtoavaimesta toimitetaan yritykselle postitse tai tekstiviestillä. Siirtoavainta tarvitsee ensimmäisen varmenteen luonnissa ja käyttäjätunnusta jokaisessa palvelupyynnössä. Onnistuneen varmenteen luonnin jälkeen yritys voi aloittaa aineistojen noutamisen ja lähettämisen pankkiin. (OP-Pohjola-ryhmä 2011.)

3.2.1 Palvelupyyntö

OP-pohjolan Web Services -yhteyskäytännössä on 4 eri palvelupyyntöä yrityksen käytettäväksi.

- downloadFileList
- downloadFile
- uploadFile
- deleteFile

Jokaista palvelupyyntöä varten tarvitsee voimassa olevan varmenteen sekä käyttäjätunnuksen. Jokainen palvelupyyntö sisältää SOAP-lähetyksessä käytettävän RequestHeader- sekä ApplicationRequest-elementin. RequestHeader on jokaisessa palvelupyynnössä rakenteeltaan samanlainen ja sisältää käyttäjätunnuksen (SenderId), uniikin palvelupyyntönumeron (RequestId), päivämäärän ja kelloajan (Timestamp), yrityksen kielen (Language) sekä vastaanottajan (ReceiverId). Koodissa 5 on esimerkki RequestHeader-elementistä. ApplicationRequest-

elementti on base64-koodattu sisältö palvelupyynnöstä. Koodissa 5 ApplicationRequest-elementin base64-koodattu sisältö on lyhennetty kolmella pisteellä luettavuuden helpottamiseksi. ApplicationRequest-elementin sisältö vaihtelee riippuen palvelupyynnöstä, mutta jokaiselle yhteiset elementit ovat käyttäjätunnus (CustomerId), päivämäärä ja kelloaika (Timestamp), ympäristö (Environment) sekä palvelupyynnön tehneen ohjelmiston nimi (SoftwareId). Lisäksi jokainen palvelupyyntö tulee digitaalisesti allekirjoittaa, joka lisää ApplicationRequest-elementin sisälle Signature-elementin. Koodissa 6 on downloadFileList-palvelupyynnön ApplicationRequest-elementin sisältö. (OP-Pohjola-ryhmä 2011.)

```
<cor:downloadFileListin xmlns:cor="http://bxd.fi/CorporateFileService">
   <mod:RequestHeader xmlns:mod="http://model.bxd.fi">
      <mod:SenderId>1000000000</mod:SenderId>
      <mod:RequestId>1313494952760</mod:RequestId>
      <mod:Timestamp>2011-08-16T14:42:28.031+03:00</mod:Timestamp>
      <mod:Language>FI</mod:Language>
      <mod:UserAgent>OP Client</mod:UserAgent>
      <mod:ReceiverId>OKOYFIHH</mod:ReceiverId>
   </mod:RequestHeader>
   <mod:ApplicationRequest
   xmlns:mod="http://model.bxd.fi">PD94bWwg...ZXF1ZXN0Pg==</mod:ApplicationRequest>
</cor:downloadFileListin>
```

Koodi 5. downloadFileList-palvelupyyntö (OP-Pohjola-ryhmä 2011).

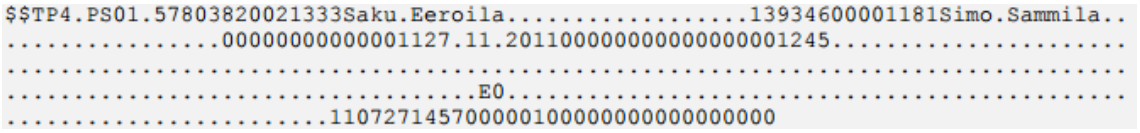
DownloadFileList-palvelupyynnöllä voidaan noutaa WS-kanavasta listaus noudettavista aineistoista. Noudettavissa on erityyppisiä aineistoja, mikä tulee huomioida jo palvelupyyntöä rakentaessa. Aineistotyypit voidaan jakaa eräaineistoihin sekä ajantasakyselyihin. Eräaineistot ovat tiliotteet, uusintatiliotteet, konsernitiliotteet, viitepalvelun tapahtumaluettelo sekä saapuvien ulkomaanmaksujen ennakkotiedot. Ajantasakyselyihin kuuluu useita saldo- ja tapahtumakyselyjä, joita voidaan käyttää muun muassa yksittäisen tai koko konsernin tilin saldon kyselyyn. Aineistotyyppi tulee ilmoittaa ApplicationRequest-elementin sisällä olevalla FileType-elementillä. Lisäksi pyyntöä voi tarkentaa Status- sekä StartDate- ja EndDate-elementeillä. Status-elementillä pyynnön voi rajata lataamattomiin, ladattuihin tai kaikkiin noudettavissa oleviin aineistoihin. StartDate- ja EndDate-elementeillä aineistopyynnön voi rajata koskemaan vain tiettyä aikaväliä. (OP-Pohjola-ryhmä 2011.)

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationRequest xmlns="http://bxd.fi/xmldata/">
   <CustomerId>1000000000</CustomerId>
   <Timestamp>2011-08-15T09:48:31.177+03:00</Timestamp>
   <Status>NEW</Status>
   <Environment>TEST</Environment>
   <SoftwareId>soft</SoftwareId>
   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
         <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
         20010315#WithComments"/>
         <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
         <Reference URI="">
            <Transforms>
               <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>sPNzEb+Mf5dchY5MTGq7GL1grEg=</DigestValue>
         </Reference>
      </SignedInfo>
      <SignatureValue>aIqreFNkxuy...nM4SXE8g==</SignatureValue>
      <KeyInfo>
         <X509Data>
            <X509Certificate>MIIC9TCCA...Iv3xpHPU=</X509Certificate>
         </X509Data>
      </KeyInfo>
   </Signature>
</ApplicationRequest>
```

Koodi 6. downloadFileList-palvelupyynnön ApplicationRequest-elementti (OP-Pohjola-ryhmä 2011).

DownloadFile-palvelupyynnöllä noudetaan haluttu aineisto. Määrittely tapahtuu aineiston tunnisteella (FileReference), jonka saa selville downloadFileList-palvelupyynnöllä. Tunniste sijoitetaan ApplicationRequest-elementin sisälle FileReferences-elementtiin. Vaikka FileReferences-elementti rakenteeltaan tukee monen tunnisteen sijoittamista yhteen sanomaan, vain yhden aineiston voi noutaa kerrallaan. (OP-Pohjola-ryhmä 2011.)

UploadFile-palvelupyyntöä käytetään aineiston lähettämiseen pankkiin. Aineisto voi olla joko merkkijono tai XML-dokumentti riippuen aineistotyypistä. Maksuaineisto lisätään ApplicationRequest-elementin Content-elementtiin base64-koodattuna. Aineiston lisäksi tulee ilmoittaa aineistotyyppi (FileType) sekä onko aineisto pakattu. Kuvassa 2 on esimerkki pikamaksun sisällöstä. Merkkijonon välilyönnit ovat korvattu pisteillä lukemisen helpottamiseksi. (OP-Pohjola-ryhmä 2011.)



Kuva 2. Pyyntö pikamaksusta (OP-Pohjola-ryhmä 2011).

WS-kanavaan lähetettyä aineistoa voi poistaa käyttämällä deleteFile-palvelupyyntöä. Tämä estää aineiston viemisen käsittelyyn ja on hyödyllinen tapa kumota mahdolliset virheet. Aineiston poistaminen tulee tehdä 30 minuuttia lähettämisen jälkeen, sillä jo käsittelyyn laitettua aineistoa ei voi enää poistaa tai peruuttaa. Jo käsittelyyn viedyn aineiston poistoyritys palauttaa vain virheilmoituksen. (OP-Pohjola-ryhmä 2011.)

3.2.2 Palveluvastaus

Jokaista palvelupyyntöä kohti WS-kanava palauttaa palveluvastauksen. Palveluvastaus seuraa samaa rakennetta kuin palvelupyynnöt, mutta RequestHeader-elementin tilalla on ResponseHeader-elementti ja ApplicationRequest-elementin tilalla ApplicationResponse-elementti. (OP-Pohjola-ryhmä 2011.)

ResponseHeader-elementti sisältää käyttäjätunnuksen, palvelupyyntönumeron ja päivämäärän ja kelloajan lisäksi vastauskoodin (ResponseCode) sekä vastaustekstin. Palvelupyynnön ollessa onnistunut WS-kanava palauttaa koodin 00 ja muissa tapauksissa virhekoodin ja sitä kuvaavan viestin. Liitteessä 1 on taulukko käytössä olevista vastauskoodeista sekä niitä vastaavat viestit. Virhekoodit ovat Nordean, OP-Pohjola-ryhmän sekä Sampo Pankin yhteisesti sopimat ja samoja kaikille pankille. (OP-Pohjola-ryhmä 2011.)

ApplicationResponse-elementin sisältö vaihtelee palvelupyynnön mukaan. DeleteFile-pyynnössä se ei sisällä allekirjoituksen lisäksi muuta kuin vastauskoodin ja -tekstin ja uploadFile -pyynnössä vain luodun aineiston tunnisteen. DownloadFileList-pyynnössä se sisältää myös listan noudettavista aineistoista. DownloadFile vuorostaan palauttaa tiedon aineistosta ja pakkauksesta sekä itse aineiston.

Vastauksen saatua ohjelmiston tulee suorittaa aineiston käsittely halutulla tavalla. (OP-Pohjola-ryhmä 2011.)

3.3 Tietoturva

WS-kanavan tietoturva perustuu salattuun tietoliikenneyhteyteen sekä julkisen avaimen infrastruktuuriin, eli PKI-menetelmään. Julkisen avaimen infrastruktuuri toteutetaan pankin sekä yrityksen välille. Pankki myöntää yritykselle tunnistautumisessa varmenteen, jota myöhemmin käytetään jokaisen lähetyksen allekirjoittamisessa. (Finanssialan Keskusliitto 2008.)

3.3.1 PKI

PKI eli Public Key Infrastructurella tarkoitetaan digitaalisten sertifikaattien hallinnoimista. Ratkaisu tarjoaa eri osapuolille mahdollisuuden tunnistaa toisensa käyttämällä yhteistä avainparia. Tunnistaminen tapahtuu julkisen avaimen sekä digitaalisen allekirjoituksen perusteella. Tavoitteena PKI:llä on helpottaa turvallista sähköistä tiedonsiirtoa, kuten sähköistä kaupankäyntiä sekä verkkopankkeja. Web Services -palvelussa käyttäjän tunnistaminen perustuu PKI-ratkaisuihin. Tärkeimpiä sertifikaatin sisältämiä ja vaadittuja tietoja ovat. (Kerttula 2000, 357.)

- Sertifikaatin myöntäjä
- Digitaalinen allekirjoitus
- Julkinen avain
- Kenelle sertifikaatti on myönnetty
- Sertifikaatin voimassaoloaika

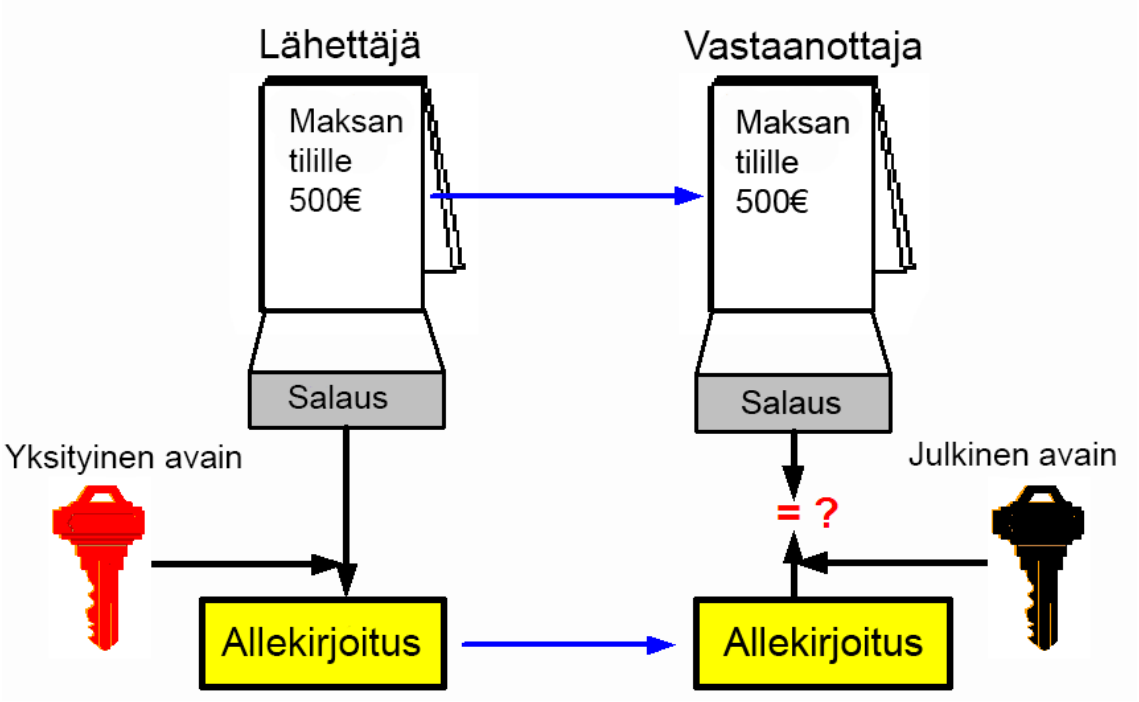
Sertifikaatin myöntäjällä tarkoitetaan organisaatiota, joka on sertifikaatin myöntänyt. Myöntäjä voi olla ketä tahansa, mutta käytettävyyden kannalta tämä tieto on tärkeää. Käyttäjien tulee miettiä luottaako ympäristö sertifikaatin myöntäjään. Niin kauan kuin käyttäjät luottavat sertifikaatin myöntäjään ja sen sertifikaattien myöntö- ja ylläpitopolitiikkaan, he voivat myös luottaa sertifikaatin myöntäjän ser-

tifikaatteihin. Verkkosivustojen HTTPS-suojauksessa tunniste ostetaan luotettavalta taholta, mutta pankkien Web Services toteutuksissa sertifikaatteja jakavat pankit itse. (Kerttula, 2000 359.)

Digitaalinen allekirjoitus on PKI-allekirjoitus, jota käytetään viestien allekirjoittamiseen. Allekirjoitus varmentaa viestin sisällön sekä allekirjoittajan henkilöllisyyden. Allekirjoitus luodaan käyttämällä omaa yksityistä avainta. Käytetty allekirjoitus voidaan myöhemmin yhdistää kyseessä olevaan varmenteeseen ja varmistua, että viestiä ei ole myöhemmin muutettu ja se on tullut vahvistetulta osapuolelta. Jos viestiä on muutettu, allekirjoitus ei enää täsmää. (Kerttula, 2000 359 – 360.)

Digitaalista allekirjoitusta tehdessään allekirjoittaja laskee viestistä yksisuuntaisen funktion avulla tiivisteen, joka salataan käyttämällä yksityistä avainta. Tiivisteellä tarkoitetaan tiedon tiivistämistä pienempään tilaan sitä varten, että alkuperäistä tietoa voidaan vertailla vertailemalla niiden tiivisteitä. Vastaanottaja laskee saamastaan viestistä aiemmin luodun tiivisteen ja vertaa sitä allekirjoittajalta saatuun tiivisteeseen. Mikäli tiivisteet ovat samat, on viesti pysynyt muuttumattomana ja sen on allekirjoittanut vahvistettu osapuoli. Kuvassa 3 on havainnollistettu esimerkki yksityisen ja julkisen avaimen käytössä viestin allekirjoittamisessa. (Kerttula, 2000, 289 – 293.)

Jokaista sertifikaattia kohden on olemassa avainpari joka sisältää salaisen sekä julkisen avaimen. Julkisen avaimen menetelmässä molemmilla osapuolilla on kaksi avainta, salainen ja julkinen. Toisin kuin salainen avain, julkista avainta saa levittää julkisesti kaikille, joiden kanssa on tarpeen vaihtaa tietoa julkisesti. (Kerttula, 2000, 359 – 363.)



Kuva 3. Digitaalinen allekirjoitus.

Sertifikaatin tiedoista saa selville kenelle sertifikaatti on myönnetty sekä sertifikaatin voimassaoloaika. Se voi sisältää käyttäjän organisaation, maan, osa-valtion tai nimen. Kaikkia tietoja ei sertifikaatin luonnissa aina tarvita. Sertifikaatille asetetaan myös voimassaoloaika, joka kertoo kuinka kauan sitä voidaan käyttää, ennen kuin se on uusittava. Jos sertifikaatti halutaan poistaa ennen voimassaoloajan päättymistä, tulee se revokoida. Tämä tarkoittaa, että sertifikaatin myöntäjä lisää sen palvelimellaan sertifikaatin revokaatiolistalle ja hylkää yritykset käyttää kyseistä sertifikaattia. (Kerttula, 2000, 359 – 363.)

3.3.2 OP-Pohjola tunnistepalvelu

OP-Pohjola on sertifikaattien myöntäjä ja tunnistepalvelun tehtävänä on tuottaa ja hallinnoida niitä varmenteita, joita käytetään WS-kanavan allekirjoitusten tarkistamisessa. Tunnistautumisen ensimmäisessä vaiheessa asiakkaan on käytävä pankissa, jotta WS-kanavan käyttäjätunnus ja siirtoavain voidaan luovuttaa vahvistetulle osapuolelle. Asiakas saa ensimmäisen osan siirtoavaimesta sopi-

musta allekirjoittaessa ja toisen osan joko tekstiviestillä tai postitse. Tämän jälkeen yritys voi luoda itselleen yksityisen avaimen sekä varmennepyynnön ja aloittaa varmenteen muodostusprosessin. (OP-Pohjola-ryhmä 2011.)

Yksityisen avaimen pituus on 2 048 bit ja algoritmi on RSA. Allekirjoituksen tiivistealgoritmi on SHA-1. (OP-Pohjola-ryhmä 2011.)

Varmennepyyntöä luodessa tulee subjektiin syöttää kaksi tietoa.

- Country (maa) = FI
- CommonName (tunnus) = WS-kanavan käyttäjätunnus

Luotu DER-muotoinen varmennepyyntö lisätään XML-dokumentin Application-Request-elementin sisälle Content-elementtiin base64-koodattuna. Lisäksi pankista saatua siirtoavainta käytetään TransferKey-elementissä. Onnistunut varmennepyyntö palauttaa asiakkaalle julkisen avaimen, jota tulee käyttää XML-viestien digitaalisessa allekirjoituksessa. (OP-Pohjola-ryhmä 2011.)

Sertifikaatti on voimassa 2 vuoden ajan, jota ennen sertifikaatti tulee uusia tai koko prosessin joutuu aloittamaan alusta uudestaan. Sertifikaatin uusimisessa luodaan uusi avainpari, jonka varmennepyyntö lähetetään pankkiin. Sertifikaatin voi uusia aikaisintaan 60 päivää ennen päättymistään. (OP-Pohjola-ryhmä 2011.)

3.4 WSDL

WSDL eli Web Service Description Language on XML:ään perustuva kieli, jota käytetään kuvaamaan tietoverkoissa tarjolla olevia WWW-teknologioihin perustuvia palveluja, eli Web Services-palveluja. WSDL-tiedostossa on tietokoneluettavia sääntöjä miten SOAP-yhteyttä tulee käyttää. Säännöt ovat esimerkiksi lista käytettävissä olevista funktioista sekä niiden parametreista ja parametrien tyypeistä. (W3C 2001.)

WSDL-dokumentti koostuu neljästä pääosasta: types, inferface, binding sekä service. Pääosat kuuluvat dokumentin description-juurielementin sisälle. Taulu-

kossa 1 on kuvattuna WSDLn eri osat. Suomen pankkien Web Services -yhteyksissä käytetään kaikille pankeille yhtenäistä WSDL-dokumenttia. (Tampereen teknillinen yliopisto 2015.)

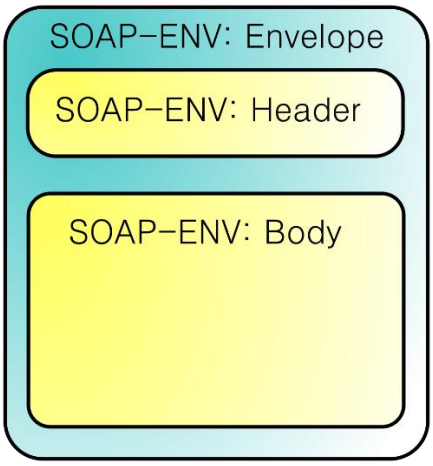
Taulukko 1. WSDL-dokumentin elementit.

Elementti	*Käännös*	*Selite*
Description	Kuvaus	WSDL-dokumentin juurielementti, jonka sisälle elementit sijoittuvat.
Types	Datatyypit	Kuvaa millaisia viestejä palvelu lähettää ja vastaanottaa.
Interface	Rajapinta	Kuvaa minkä abstraktin toiminnallisuuden Web-palvelu tarjoaa.
Binding	Sidos	Kuvaa miten palveluun voidaan ottaa yhteyttä, eli sitoo määritellyt operaatiot ja viestit todellisiin protokolliin ja sanomatyyppiin..
Service	Palvelu	Kuvaa missä palvelu sijaitsee, eli palveluun kuuluvat portit.

3.5 Simple Object Access Protocol

SOAP eli Simple Object Access Protocol on riippumaton tietoliikenneprotokolla, jonka pääasiallisena tehtävänä on pyyntö-vaste -kommunikointi eli proseduurinen etäkutsu. SOAP-kirjekuori kuljettaa dataa ja sitä käytetään pääasiassa http-protokollan yli. SOAP soveltuukin hyvin Internetiä hyödyntäviin B2B- ja B2C-sovelluksiin ja sen keveyden vuoksi myös pienille laitteille, joissa on vain http-ympäristö ja XML-jäsennin. SOAPin tärkeimpänä etuna on sen laajennettavuus. Viestiin voidaan sisällyttää esimerkiksi sovellusriippumatonta informaatiota. Web Services -yhteyksissä kaikki lähetettävä ja vastaanotettava tieto on SOAP-kirjekuoren sisällä. (Tampereen teknillinen yliopisto 2014.)

SOAP-viestin juurielementti on kirjekuori (Envelope), jonka sisällä ovat muut SOAP-elementit. Muita SOAP-viestin elementtejä ovat Header-, Body- sekä Fault-elementit. Kuvassa 4 on esitetty SOAP-viestin elementit ja viestin rakenne. Nämä elementit saavat jokainen esiintyä vain yhden kerran. SOAP pohjautuu XML-kieleen. Kuten XML, SOAP-viesti alkaa juurielementillä (Envelope) tai mahdollisesti XML-prologilla. (Tampereen teknillinen yliopisto 2014.)



Kuva 4. SOAP-kirjekuoren rakenne (Wikipedia 2013).

SOAP-viestin Header-elementti määrittelee kirjekuoren otsikkotiedot. Otsikkomerkinnät eivät seuraa standardeja, joten otsikon sisältö on sovelluskohtaista. Pankkien Web Services lähetyksissä otsikkoon sijoitetaan useita allekirjoituksia sekä tiivisteitä. (Tampereen teknillinen yliopisto 2014.)

SOAP-viestin Body, eli runko on aina pakollinen osa SOAP-viestiä. Elementillä välitetään haluttua XML-muotoista tietoa tai virheilmoituksia. Standardit eivät kuitenkaan määrittele Body-elementin sisältöä, minkä vuoksi kaikille elementeille tulisi määrittää nimiavaruus. Taulukossa 2 on yksinkertaistettu SOAP-kirjekuoren elementit ja niiden pakollisuus. (Tampereen teknillinen yliopisto 2014.)

Taulukko 2. SOAP-elementit.

Elementti	*Kuvaus*	*Pakollinen*
Envelope	Tunnistaa XML-dokumentin SOAP-elementiksi.	Kyllä
Header	Sisältää otsikkomerkinnät.	Ei
Body	Sisältää kutsun ja XML-muotoista tietoa.	Kyllä
Fault	Sisältää tietoa mahdollisista virheistä joita syntyi viestin prosessoinnin aikana.	Ei

4 WEB SERVICES -PALVELUN KÄYTTÖÖNOTTO

Tehden-ohjelmistoon toteutettu Web Services -palvelu koostuu 4:stä eri osasta: lähetettävän aineiston luonti, vastaanotetun aineiston käsittely, varmenteiden ja avaimien luonti sekä SOAP-yhteydet ja lähetyksien allekirjoitus. Web Services -palvelua käytetään viitesiirtoaineistojen automaattiseen sekä tarvittaessa manuaaliseen noutoon pankista. Aineiston noutamiseen vaaditaan voimassaoleva varmenne sekä käyttäjätunnus.

4.1 Luokkarakenne

Tehden-ohjelmiston Web Services -toteutus muodostuu 5 eri luokasta sekä kahdesta käyttöön otetusta kirjastosta. Luokkarakenne on luotu eri käyttövaiheiden mukaiseksi: OpApplicationRequest- (lähetettävän aineiston luonti), OpApplicationResponse- (vastaanotetun aineiston käsittely), SoapClient- (SOAP-yhteyden muodostus ja aineiston lähetys), WebServicesModel- (Tehden-ohjelmiston tietokannan käsittely) sekä BankWebService-luokka, joka toimii kaikkien näiden luokkien välillä.

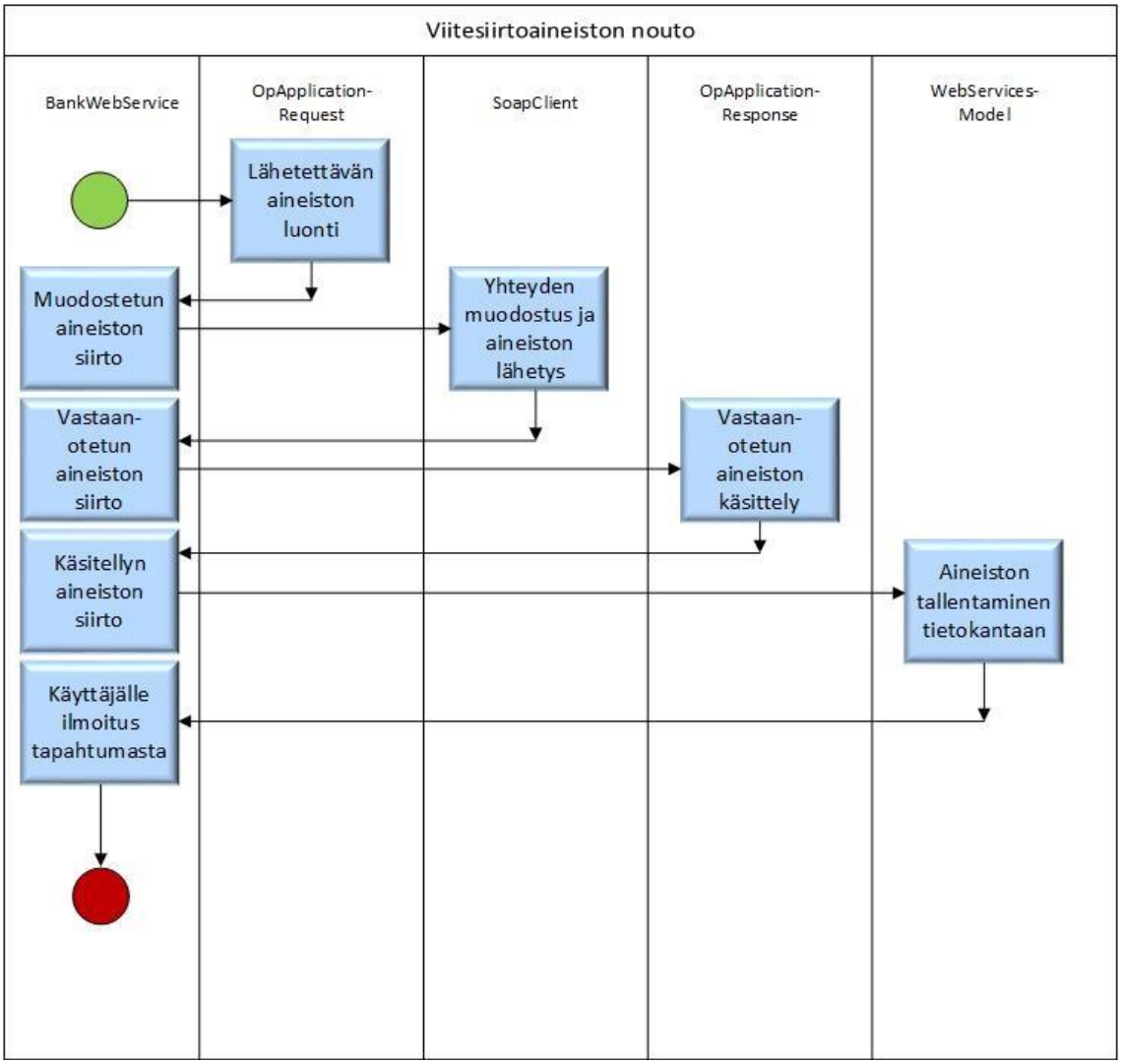
BankWebService-luokka helpottaa Web Services -palvelun käyttöä. Luokkaa alustaessa se lataa määritetylle pankille kuuluvat WSDL-tiedostot sekä muut pankkikohtaiset asetukset. Alustuksen jälkeen luokan funktioita voidaan kutsua mistä tahansa osiosta ohjelmaa. Koodissa 7 on esimerkki viitesiirtoaineiston noudosta käyttäen WS-kanavan käyttäjätunnusta sekä yksityistä ja julkista avainta.

```
$this->load->library('bankwebservice');
$payments = $this->bankwebservice->fetchPaymentInformation($username, $privatekey, $publickey);
```

Koodi 7. Viitesiirtoaineiston nouto.

BankWebService-luokka käyttää OpApplicationRequest-, OpApplicationResponse- sekä SoapClient-apuluokkia aineiston luonnissa, käsittelyssä sekä lähet-

tämisessä. Esimerkiksi tiedoston noudossa pyyntösanoma luodaan opApplicationRequest-luokassa, lähetetään ja allekirjoitetaan SoapClient-luokassa ja vastaanotettu aineisto käsitellään OpApplicationResponse-luokassa. Käsitelty ja varmennettu aineisto lähetetään WebServicesModel-luokkaan, jossa se tallennetaan tietokantaan. Kuvassa 5 on kaaviorakenne Tehden-ohjelmiston viitesiirtoaineiston noudon prosesseista ja käytetyistä luokista.



Kuva 5. Viitesiirtoaineiston nouto Tehden-ohjelmistossa.

4.2 Sertifikaatin luonti ja varmentaminen

Web Services -yhteyskäytännön toteutuksessa käytetään PHP:n OpenSSL- kirjastoa generateCertificateAndPrivateKey-funktiossa uuden avainparin luomiseen

(Liite 2) sekä vanhan uusimiseen. Funktio luo käyttäjätunnuksella PEM-muotoisen sertifikaattipyynnön sekä yksityisen avaimen. Sertifikaattipyyntö sekä avain tallennetaan salattuna tietokantaan varmennepyyntöä varten.

Ensimmäistä varmennepyyntöä luotaessa käytetään tallennettua sertifikaattipyyntöä, käyttäjätunnusta sekä pankista saatua siirtoavainta. Sertifikaattipyyntö base64-koodataan ja sijoitetaan lähetettävän XML:n ApplicationRequest-elementtiin. Myös käyttäjätunnus ja siirtoavain sijoitetaan omiin elementteihin. XML-sanoma muodostetaan käyttämällä PHP:n SimpleXMLElement-luokkaa. Koodissa 8 on esimerkki varmennepyynnön XML-dokumentin luonnista.

```
$xml = new SimpleXMLElement(
    '<?xml version="1.0" encoding="UTF-8">
    <CertApplicationRequest xmlns="http://op.fi/mlp/xmldata">
    </CertApplicationRequest>'
);
$xml->addChild('CustomerId', $username);
$xml->addChild('Timestamp', date('c'));
$xml->addChild('Environment', $this->environment);
$xml->addChild('SoftwareId', 'Tehden');
$xml->addChild('Compression', 'false');
$xml->addChild('Service', 'MATU');
$xml->addChild('Content', base64_decode($certificate));
$xml->addChild('TransferKey', $transferkey);
```

Koodi 8. Varmennepyynnön XML-dokumentin luonti.

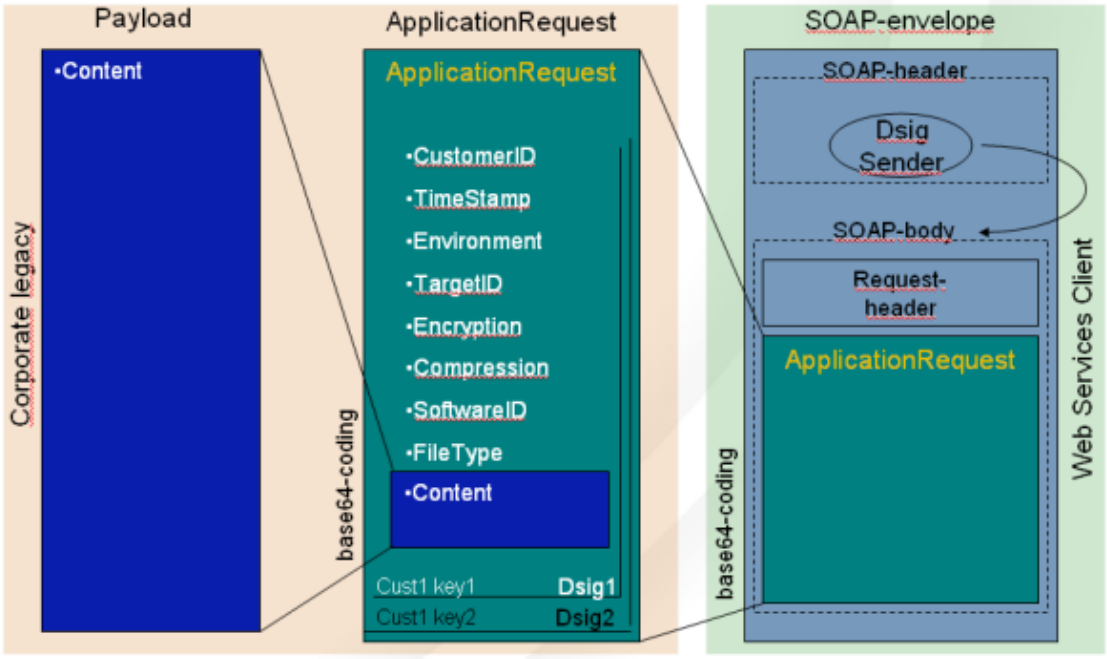
Luotu ApplicationRequest-elementti base64-koodataan ja sijoitetaan SOAP-kirjekuoren Body-elementtiin. Ensimmäistä varmennepyyntöä tehdessä lähetettävää aineistoa ei tarvitse allekirjoittaa. Onnistuneen varmennepyynnön jälkeen WS-kanava palauttaa asiakkaalle luodun sertifikaatin, joka on voimassa kaksi vuotta. Tätä sertifikaattia, eli julkista avainta, käytetään jatkossa jokaisen aineiston allekirjoituksessa yksityisen avaimen kanssa.

Varmenteen uusiminen tehdään aikaisintaan 60 kalenteripäivää ennen edellisen varmenteen vanhenemista samalla tavalla kuin uuden luominen, paitsi pankista saatua siirtoavainta ei käytetä. Varmennus tapahtuu allekirjoittamalla pyyntö käyttämällä vanhentuvia varmenteita. OP-Pohjolan Web Services -testipuolella varmennetta uusittaessa myös vanha varmenne pysyi käytettävissä.

4.3 Sanoman muodostus aineiston noutoon

Viitesiirtoaineistoa noudettaessa muodostetaan allekirjoitettu SOAP-pyyntösanoma ja käyttäen http-protokollaa lähetetään se oikeaan verkko-osoitteeseen. Pyyntösanomaa lähettäessä jäädään odottamaan vastausta, jonka sisältö voidaan tulkita ISO 20022 -standardin mukaisesti. Pyyntösanoma muodostetaan ja lähetetään kuvan 6 mukaisesti ja sen vaiheet ovat seuraavat:

1. Luodaan objekti ApplicationHeader- sekä ApplicationRequest-elementeistä.
2. Lisätään ApplicationHeader-elementtiin vaadittavat lapsielementit.
3. Luodaan ApplicationRequest XML-dokumentti.
4. Lisätään XML-dokumenttiin vaadittavat lapsielementit.
5. Digitaalisesti allekirjoitetaan ApplicationRequest-elementti ja lasketaan tiivisteet käyttämällä yksityistä avainta.
6. Allekirjoituksen jälkeen Base64-koodataan ApplicationRequest-elementti.
7. Luodaan WSDL:n määrittelyjen mukainen SOAP-sanoma.
8. Sijoitetaan ApplicationHeader- sekä ApplicationRequest-elementit SOAP-sanoman runkoon.
9. Digitaalisesti allekirjoitetaan SOAP-otsikko ja lasketaan tiivisteet käyttämällä yksityistä avainta.
10. Lähetetään SOAP-sanoma WSDL:ssä kuvattuun osoitteeseen ja odotetaan vastausta.



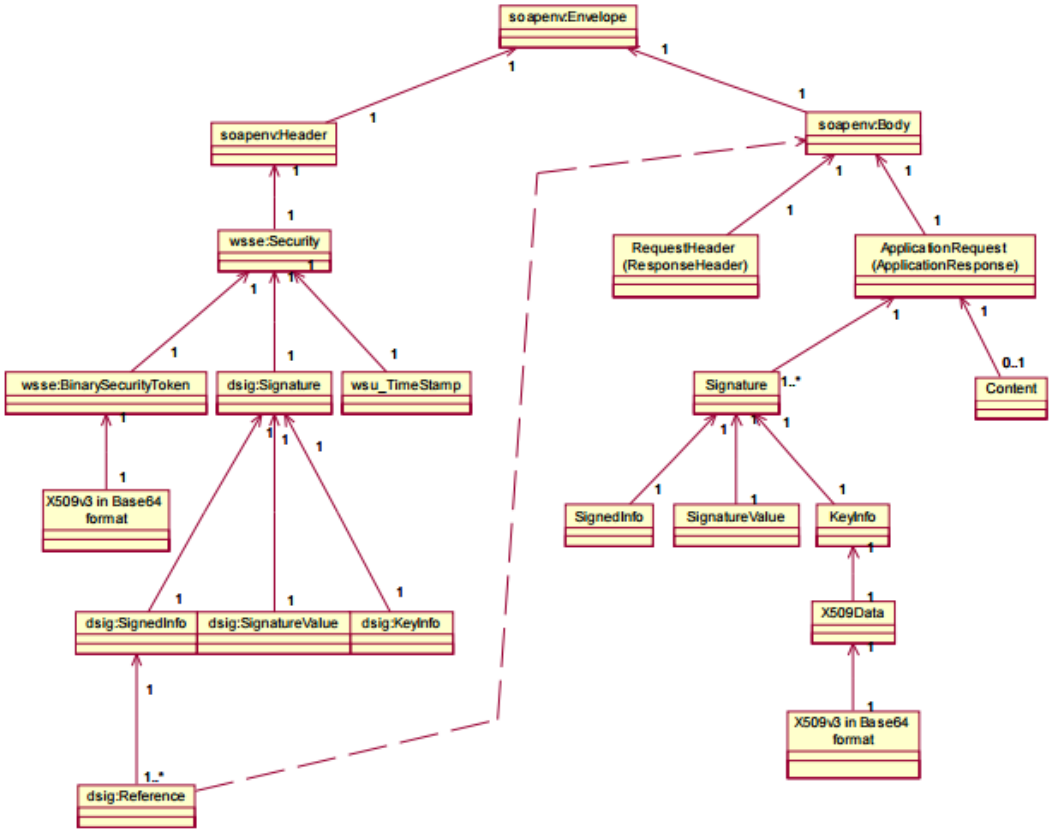
Kuva 6. Web Services -pyyntösanoma (Samlink 2014).

4.4 Digitaalinen allekirjoitus

WS-kanavassa sanomien ja palvelupyyntöjen muuttumattomuuden ja aitouden varmistaminen perustuu digitaaliseen allekirjoitukseen. WS-kanavassa käytetään kaksitasoista allekirjoitusta, jossa lähetettävä palvelupyyntö (ApplicationRequest) sekä SOAP-kirjekuori allekirjoitetaan. Pyyntö allekirjoitetaan, että vastaanottaja voi varmistua palvelupyynnön tulleen todennetulta henkilöltä. Avustuksena digitaaliseen allekirjoitukseen käytetään XmlDSig-kirjastoa.

ApplicationRequest-elementti allekirjoitetaan lisäämällä palvelupyyntöön Signature-elementti, joka sisältää digitaalisen allekirjoituksen, lasketut tiivisteet sekä julkisen avaimen. Allekirjoitus on Base64-koodattu merkkijono, joka lasketaan ja luodaan käyttämällä yksityistä avainta. Allekirjoituksessa lisäksi tiivistetään ApplicationRequest-elementin alkuperäinen sisältö käyttämällä SHA-1 salausta sekä c14-kanonikalisointialgoritmia. Laskettu tiiviste lisätään DigestValue-elementtiin. Luodun allekirjoituksen aitouden voi vahvistaa käyttämällä avainpariin

kuuluvaa julkista avainta. Julkinen avain lisätään KeyInfo-elementin sisälle varmenteen tarkistamiseksi. Kuvassa 7 on ApplicationRequest-elementin alla allekirjoitukseen käytetyt eri elementit.



Kuva 7. ApplicationRequest-elementti (Finanssialan Keskusliitto 2008).

SOAP-kirjekuoressa allekirjoitus ja tiivisteet lasketaan käyttämällä samoja allekirjoitustekniikoita sekä kanonikalisointialgoritmeja. Allekirjoitus luodaan uudestaan käyttämällä yksityistä avainta ja julkinen avain lisätään BinarySecurityToken-elementin sisälle. Kirjekuoreen lisätään lisäksi TimeStamp-elementti, josta luodaan oma tiiviste. TimeStamp-elementti sisältää tiedon milloin SOAP-kirjekuori on luotu ja mihin asti se on voimassa. Kirjekuoren toinen tiiviste lasketaan SOAP-rungosta ja laskettu tiiviste lisätään SOAP-otsikkoon.

4.5 Aineiston vastaanotto ja tallennus

Aineistoa noudettaessa ensin tehdään downloadFileList-palvelupyyntö, joka palauttaa aineistotyypin mukaiset tiedot ladattavista tiedostoista. Listasta valitaan FileReference-elementit, joita käyttämällä tehdään downloadFile-palvelupyyntö. DownloadFile voidaan kutsua vain yhdellä FileReference-elementillä kerrallaan, joten usein samaa funktiota tulee kutsua monta kertaa.

Aineiston noudossa voi päättää noudettavan aineiston muodon. Vaihtoehtoina viitesiirtoaineistossa on perinteinen viitepalveluntapahtumaluettelo merkkijonona sekä ISO20022-standardiin perustuva XML-dokumentti. Koodissa 9 on kuvattuna standardin mukainen tilitapahtuma XML-tiedostossa. Sama tieto perinteisenä merkkijonona on kuvan 8 merkkijono.

```
3571004202631001507271507240727 5UTZ000659660000000000000000012345YRITYS OY 1 00000257100J
```

Kuva 8. Viitetapahtuma merkkijonona.

Viitetapahtumaluettelossa jokaiselle tiedolle on varattu oma määrä mahdollisia merkkejä. Esimerkiksi tilinumeron pituus on 14 merkkiä, joka alkaa merkkijonon toisesta merkistä. Yhden tapahtumatietueen pituus on 90 merkkiä, joka sisältää samat tiedot kuin standardin mukainen XML-dokumentti.

```
<TxDtls>
<Refs>
    <AcctSvcrRef>SCT003S9N4CUD1B</AcctSvcrRef>
</Refs>
<AmtDtls>
    <TxAmt>
        <Amt Ccy="EUR">257.10</Amt>
    </TxAmt>
</AmtDtls>
<RltdPties>
    <Dbtr>
        <Nm>YRITYS OY</Nm>
    </Dbtr>
</RltdPties>
<RmtInf>
    <Strd>
        <CdtrRefInf>
            <Tp>
                <CdOrPrtry>
                    <Cd>SCOR</Cd>
                </CdOrPrtry>
            </Tp>
            <Ref>000000000000000012345</Ref>
        </CdtrRefInf>
    </Strd>
</RmtInf>
<RltdDts>
    <AccptncDtTm>2015-07-24T03:58:00+02:00</AccptncDtTm>
</RltdDts>
</TxDtls>
```

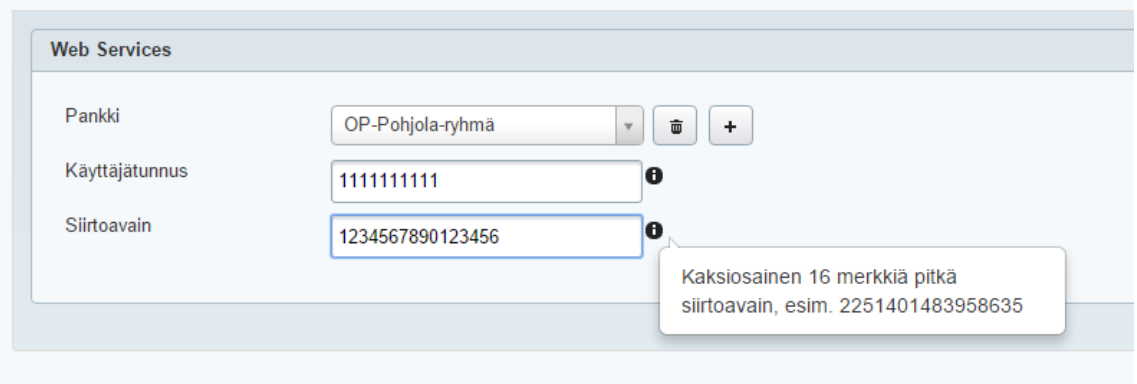
Koodi 9. ISO 20022 -standardin mukainen viitteellinen tilitapahtuma.

Aineiston käsittelyyn on omat funktiokäsittelyt sen perusteella onko vastaanotettava aineisto XML-dokumentti vai merkkijono. Molemmat mahdollisuudet haluttiin pitää, koska asiakkailla voi olla erilaisia sopimuksia pankkinsa kanssa aineiston muodosta. Molemmista kuitenkin saadaan samat tiedot, mitä käyttämällä aineiston käsittely on lähes identtistä. Liitteessä 8 on osa aineiston käsittelyyn ja kohdentamiseen käytetystä funktiosta readReferencePaymentsToAccountsReceivable. Aineiston käsittelyssä tapahtuvat eri vaiheet.yksinkertaistettuna ovat seuraavat:

1. Tarkastetaan ettei aineistoa ole jo aiemmin tuotu ohjelmistoon.
2. Tarkistetaan ettei kyseistä suoritusta haluta suodattaa pois.
3. Lisätään viitteelliset suoritukset tietokantaan.
4. Tarkistetaan onko kyseisellä viitenumerolla lähetty Tehden-ohjelmistossa lasku.
5. Kohdennetaan lähetetty lasku vastaanotettuun maksuaineistoon.
6. Lähetetään käyttäjälle yhteenveto tapahtumasta sähköpostilla

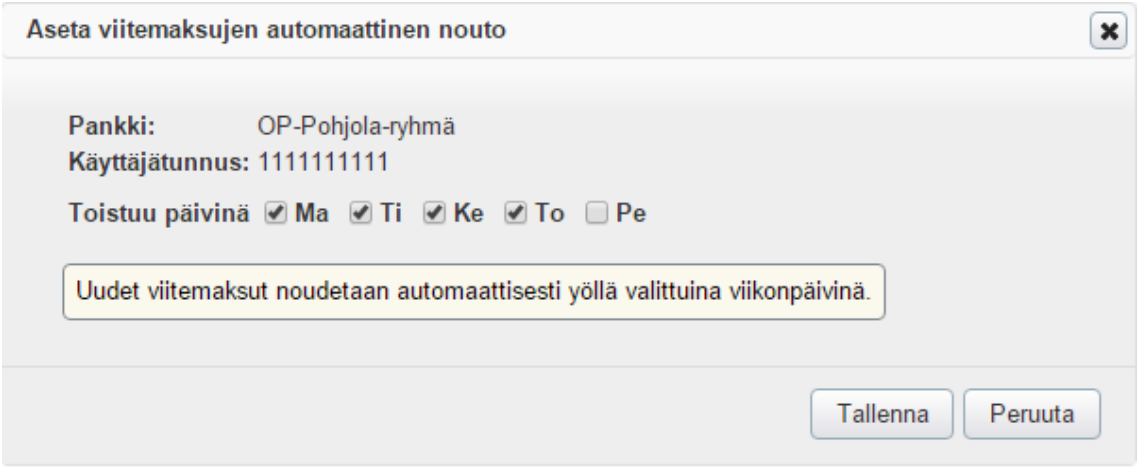
4.6 Web Services -palvelun käyttö Tehden-ohjelmistossa

Tehtyään Web Services -sopimuksen pankin kanssa, käyttäjä voi syöttää käyttäjätunnuksen sekä siirtoavaimen ohjelmaan. Kuvassa 9 on näkymä tunnusten syöttämisestä. Käyttäjän tallennettua tunnukset, ohjelma luo automaattisesti tunnuksella varmennepyynnön ja suorittaa ensimmäisen varmenteen noudon Web Services -kanavasta. Noudon jälkeen ohjelma ilmoittaa käyttäjälle oliko varmenteen nouto onnistunut. Onnistuneessa tilanteessa ohjelma opastaa käyttäjää asettamaan viitemaksuille automaattisen noudon.



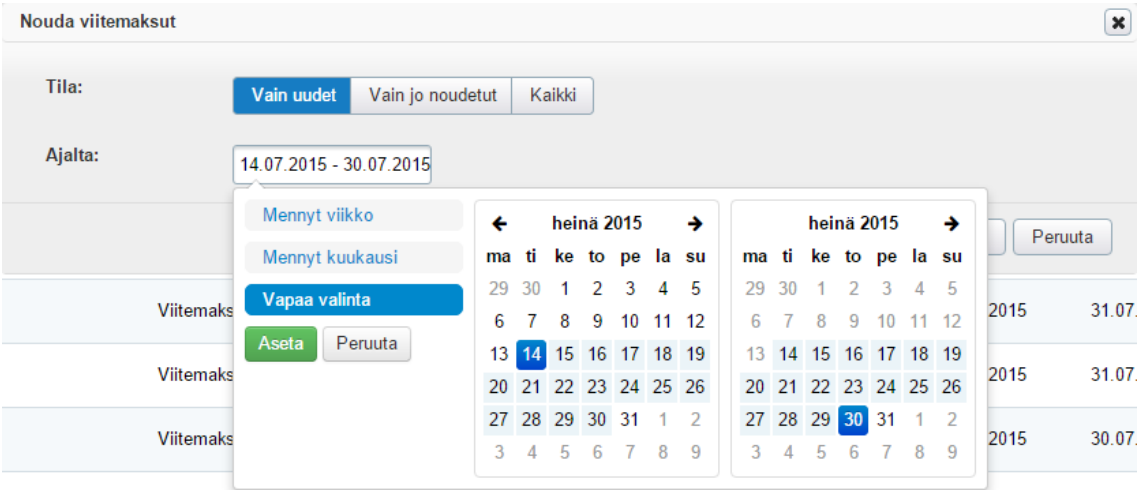
Kuva 9. Web Services -tunnusten asettaminen.

Viitesiirtoaineiston noudon voi joko automatisoida tai suorittaa manuaalisesti. Automaattisessa noudossa (Kuva 10) käyttäjä voi valita minä arkipäivinä nouto tehdään. Nouto suoritetaan iltaisin, koska Osuuspankki vastaanottaa päivän viitemaksuja klo 17.00 asti ja lista aineistoista on noudettavissa noin klo 21.30.



Kuva 10. Dialogi viitesiirtoaineiston automaattiseen noutoon.

Osuuspankilla sekä Tehdenillä voi olla iltaisin huoltotoimenpiteitä, jolloin illalla suoritettava aineistonouto ei ole aina käytettävissä. Tästä syystä myös käyttäjälle tarjotaan mahdollisuus viitemaksujen manuaaliseen noutoon. Manuaalisessa noudossa käyttäjä voi täsmentää noudettavien viitemaksujen tilan sekä miltä ajalta viitemaksut noudetaan. Kuvassa 11 on näkymä viitesiirtoaineiston manuaalisesta noudosta.



Kuva 11. Viitesiirtoaineiston manuaalinen nouto.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa Tehden-toiminnanohjausjärjestelmän ja Osuuspankin välille toimiva Web Services -yhteys, jota voidaan käyttää viitesiirtoaineiston noutoon. Toteutuksen tuli myös sisältää WS-kanavan varmenteiden luonnin, niiden hallinnoinnin sekä viitesiirtoaineiston tallentamisen tietokantaan. Ominaisuuksien luontia varten tuli olla hyvä tietämys ohjelmoinnista, XML-merkintäkielestä, julkisen avaimen infrastruktuurista, SOAP-tietoliikenneprotokollasta sekä Web Services -yhteyskäytännön toiminnasta.

Opinnäytetyön tavoitteet saavutettiin ja Tehden-ohjelmiston Web Services -palvelu on kirjoitushetkellä jo usealla yrityksellä käytössä. Monipuolinen toteutus mahdollisti käyttäjälle yksinkertaisen ja helpon käytön. Toteutus automatisoi monia prosesseja ja käyttäjälle jää työksi vain tunnusten ensimmäinen varmentaminen.

Valmis toteutus osoitti Web Services -yhteyskäytännön hyödyllisyyden pilvipalveluna toimivassa toiminnanohjausjärjestelmässä. Saman yhteyskäytännön voi skaalata jokaisen saman pankin asiakkaan tarpeisiin. Lisäksi yhteisten standardien vuoksi työtä voi helposti laajentaa toimimaan myös muiden pankkien yhteyskäytäntöjen kanssa.

Web Services -yhteyskäytäntö antaa kehittäjälle monia erilaisia mahdollisuuksia palvelun toteuttamisessa. Toteutus PHP-ohjelmointikielellä WWW-sovelluksessa on vain yksi mahdollisuus muiden joukossa. Web Services -kanava voidaan toteuttaa myös mobiili- ja tietokonesovelluksiin käyttäen useita eri ohjelmointikieliä. PHP:lla toteutettu kokonaisuus tarjoaa kehittäjälle kirjastot esimerkiksi SOAP-protokollaan sekä OpenSSL-sertifikaattien ja -avainten luomiseen. Toisin kuin .Net-ohjelmistokomponenttikirjasto, PHP ei tarjoa ominaisuutta SOAP-viestien digitaaliseen allekirjoittamiseen, vaan tämä tulee tehdä itse tai käyttää ulkopuolista kirjastoa.

Ongelmana kehityksessä olivat Osuuspankin tarjoamat dokumentit sekä kanavasta palautuvat virheviestit. Dokumentit käsittelivät kanavaa teoriatasolla, eikä

muutamaan kanavan ominaisuuteen ollut dokumentaatiota lainkaan. Lisäksi tietoa kanavan käytöstä joutui keräämään monista eri Osuuspankin lähteistä. Kanavasta palautuvat virheviestit eivät sisältäneet yksityiskohtaista tietoa tapahtuneesta vaan useat ongelmat palauttivat vain virheviestin ”Technical error”.

Opinnäytetyötä tehdessä vaatimukset ja tavoitteet pysyivät alusta loppuun samoina, mikä helpotti kehitystyötä. Myös Tehden-ohjelmiston tietokantarakenne sekä käytössä oleva sovelluskehys sopivat hyvin WS-kanavan toteutukseen. Toteutettu luokkarakenne sopii hyvin myös jatkokehitykseen.

Toteutettua Web Services -palvelua Tehden-ohjelmistossa on tarkoitus laajentaa myös muiden pankkien kanssa toimivaksi. Muiden pankkien lisääminen ohjelmistoon on helppoa pankkien yhteisten standardien vuoksi.

LÄHTEET

Finanssialan Keskusliitto 2008. Security and Message Specification for Financial Messages using Web Services. Viitattu 11.8.2015 http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/WebServices_Messages_20081022_105.pdf.

Finanssialan Keskusliitto 2010. Web Services & PKI. Viitattu 31.8.2015 https://www.fkl.fi/teemasivut/sepa/vaikutukset_yritystoimintaan/Dokumentit/Web_Services_PKI_210120101.pdf.

Finanssialan Keskusliitto 2013. Yhtenäisen Euromaksualueen toteutuminen Suomessa. Viitattu 30.8.2015 https://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/SEPA_siirtymasuunnitelma_v5.pdf.

Finanssialan Keskusliitto 2014. Maksaminen verkossa. Viitattu 11.8.2015 https://www.fkl.fi/teemasivut/pankkiturvallisuus/ammattilaiselle/Sivut/maksaminen_verkossa.aspx.

Finanssialan Keskusliitto 2015. Yhtenäiseen euromaksualueeseen SEPAan liittyvät pankkien palvelut. Viitattu 15.6.2015 https://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/Pankkien_SEPA_palvelutaulukko.pdf.

ISO 2015. The ISO 20022 Adoption Iniatives Report. Viitattu 11.8.2015 http://www.iso20022.org/documents/adoption/ISO20022_adoption_report.pdf.

Kerttula, E. 2000. Tietoverkkojen tietoturva. 3. painos. Helsinki: Oy Edita Ab.

Matti Luoto 2008. Sepa on selvää säästöä. Viitattu 24.8.2015 http://www.talouselama.fi/minavaitan/sepa+on+selvaa+saastoa/a2086375.

Microsoft 2002. Understanding XML Namespaces. Viitattu 14.8.2015 https://msdn.microsoft.com/en-us/library/aa468565.aspx.

Nordea 2015. Web Services. Viitattu 11.8.2015 http://www.nordea.fi/yritykset+ja+yhteis%C3%B6t/maksuliike/yhteys+pankkiin/web+services/1126622.html.

O'Reilly Media, Inc. 2014. What is XML?. Viitattu 1.9.2015 http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58.

OP-Palvelut Oy 2014. Yrityksen pankkiyhteys -kanavasta noudettavien XML -tiliraportointi aineistojen asiakasohje. Viitattu 17.8.2015 https://www.op.fi/media/liitteet?cid=151806738&srcpl=4.

OP-Pohjola-ryhmä 2011. Yrityksen pankkiyhteys -kanavan eli Web Services -kanavan ja sen tunnistepalvelun sovellusohje. Viitattu 18.8.2015 https://www.op.fi/media/liitteet?cid=151240134&srcpl=4.

Samlink 2013. XML Maksuluettelo. Viitattu 11.8.2015 http://www.samlink.fi/Global/Ohjelmistopalvelut/FI/XML_maksuluettelo_soveltamisohje.pdf.

Samlink 2014. Web Services -yhteys. Viitattu 1.8.2015 http://www.samlink.fi/Global/Ohjelmistopalvelut/FI/WSPalvelukuvausSHB.pdf.

Suomen Pankki 2015. Yhtenäinen euromaksualue (SEPA). Viitattu 12.8.2015 http://www.suomenpankki.fi/fi/rahoitusjarjestelman_vakaus/kehityshankkeet/pages/sepa.aspx.

Tampereen teknillinen yliopisto 2014. Simple Object Access Protocol (SOAP). Viitattu 12.8.2015 http://www.cs.tut.fi/~palpo/materiaali/luennot2014/luento2-2-soap.pdf.

Tampereen teknillinen yliopisto 2015. Web Service Description Language (WSDL). Viitattu 10.8.2015 http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/5.pdf.

Tehden Oy 2015. Edelläkävijänä pilveen. Viitattu 1.9.2015 http://www.tehden.com/yritys.

W3C 2000. Extensible Markup Language (XML) 1.0 (Second Edition). Viitattu 1.9.2015 http://www.w3.org/TR/2000/REC-xml-20001006#sec-prolog-dtd.

W3C 2001. Web Services Description Language (WSDL) 1.1. Viitattu 1.8.2015 http://www.w3.org/TR/wsdl.

W3C 2004. XML Schema. Viitattu 23.8.2015 http://www.w3.org/XML/Schema.

W3C 2014. XML in 10 points. Viitattu 19.8.2015 http://www.w3.org/XML/1999/XML-in-10-points.html.en.

W3C 2015. Extensible Markup Language (XML). Viitattu 20.8.2015 http://www.w3.org/XML/.

W3Schools 2015. XML Attributes. Viitattu 1.9.2015 http://www.w3schools.com/xml/xml_attributes.asp.

Wikipedia 2013. SOAP. Viitattu 2.9.2015 https://upload.wikimedia.org/wikipedia/commons/thumb/5/59/SOAP.svg/961px-SOAP.svg.png.

Web Services -vastausviestit

Code	Name	Remarks
00	OK.	
01	Pending.	not used
02	SOAP signature error.	signature verification failed
03	SOAP signature error.	certificate not valid for this id
04	SOAP signature error.	certificate not valid
05	Operation unknown.	
06	Operation is restricted.	
07	SenderID not found.	
08	SenderID locked.	
09	Contract locked.	
10	SenderID outdated	
11	Contract outdated	
12	Schemavalidation failed.	
13	CustomerID not found.	
14	CustomerID locked.	
15	CustomerID outdated.	
16	Product contract outdated.	
17	Product contract locked.	
18	Content digital signature not valid.	
19	Content certificate not valid.	
20	Content type not valid.	
21	Deflate error.	
22	Decrypt error.	
23	Content processing error.	
24	Content not found.	
25	Content not allowed.	
26	Technical error.	
27	Cannot be deleted.	
28	[not used]	not used
29	Invalid parameters.	
30	Authentication failed	
31	Duplicate message rejected.	SOAP.Body.RequestHeader.SenderId + SOAP.Body.ReqhestHeader.RequestId
32	Duplicate ApplicationRequest rejected.	ApplicationRequest.CustomerId + ApplicationRequest.Timestamp

generateCertificateAndPrivateKey-funktio webServicesModel-luokassa

```
/**
 * Create new certificate request and private key for getting web ser-
vice certificate
 * Is used in getting the first certificate + certificate renew
 * @param $commonName
 *
 * @return stdClass
 */
public function generateCertificateAndPrivateKey($commonName) {

    $dn = array(
        "C" => "FI",
        "CN" => $commonName
    );


    $sslConfPath = str_replace('\\', '/', realpath(AP-
PPATH).'/fakepath/openssl.cfg');

    $SSLcnf = array(
        'config' => $sslConfPath,
        'private_key_type' => OPENSSL_KEYTYPE_RSA,
        'digest_alg' => 'sha1',
        'private_key_bits' => 2048
         );

    // Generate a new private (and public) key pair
    $privkey = openssl_pkey_new($SSLcnf);


    // Generate a certificate signing request
    $csr = openssl_csr_new($dn, $privkey, $SSLcnf);

    openssl_csr_export($csr, $csrout);
    openssl_pkey_export($privkey, $pkeyout);

    $derOutPut = $this->pem2Der($csrout);

    $result = new stdClass();
    $result->derCertificate = $derOutPut;
    $result->privateKey = $pkeyout;
    if (isset($this->encryptionKey) && !empty($this->encryptionKey)) {
        $result->hashedPrivateKey = $this->general->encrypt($pkeyout,
$this->encryptionKey);
    }

    return $result;
}
```

getCertificate-funktio BankWebService-luokassa

```
/**
 * @param $username
 * @param $transferkey
 * @param $certificate
 *
 * @return mixed
 * Get certificate for web service. Used when the WS is certified for the first time (with transferkey)
 */
public function getCertificate($username, $transferkey, $certificate)
{
    $arguments = $this->CI->opapplicationrequest->getCertificateArguments($username, $transferkey, $certificate);

    $client = new SoapClient(
        $this->certService,
        array(
            'trace' => TRUE,
            'exceptions' => FALSE
        )
    );

    $return = $client->__call('getCertificate', $arguments);

    $result = $this->CI->opapplicationresponse->parseCertificate($return);

    return $result;
}
```

renewCertificate-funktio BankWebService-luokassa

```
    /**
     * @param $username
     * @param $newCertificate
     * @param $privatekey
     * @param $publickey
     *
     * @return mixed
     * Gets a new certificate for a web service. Used when old certificate is running out of time and new certificate is needed (when cert has less than 60 days)
     */
   public function renewCertificate($username, $newCertificate, $privatekey, $publickey) {
      $arguments = $this->CI->opapplicationrequest->renewCertificateArguments($username, $newCertificate, $privatekey, $publickey);

      $client = new SoapClient(
         $this->certService,
         array(
            'trace' => TRUE,
            'exceptions' => FALSE
         )
      );

      $return = $client->__call('getCertificate', $arguments);

      $result = $this->CI->opapplicationresponse->parseCertificate($return);

      return $result;
   }
```

fetchPaymentInformation-funktio BankWebService-luokassa

```
/**
 * @param $username
 * @param $privatekey
 * @param $publickey
 *
 * @return array|bool
 * Fetch payment information
 */
public function fetchPaymentInformation(
    $username,
    $privatekey,
    $publickey,
    $status = 'NEW',
    $startdate = NULL,
    $enddate = NULL){

    $fileList = $this->getFileList($username, $privatekey, $publickey,
$status, $startdate, $enddate);
    if ($fileList !== FALSE && $fileList !== NULL) {
        $downloadedFiles = array();
        foreach ($fileList as $file) {
            $downloadedFile = $this->getFile($username, $privatekey,
$publickey, $file);
            if ($downloadedFile) {
                $file->payment = $downloadedFile->payment;
                array_push($downloadedFiles, $file);
            }
        }
        return $downloadedFiles;
    }
    if ($fileList === NULL) {
        return TRUE;
    }
    return FALSE;
}
```

getFileList-funktio BankWebService-luokassa

```
/**
 * @param $username
 * @param $privatekey
 * @param $publickey
 *
 * @return bool
 * Get list of files and parses the return to get array of different
files to download
 * Filetype for WS: camt.054.001.02 TL (xml) / TL (string)
 */
public function getFileList(
    $username,
    $privatekey,
    $publickey,
    $status = 'NEW',
    $startdate = NULL,
    $enddate = NULL) {

    $arguments = $this->CI->opapplicationrequest->getArgumentsToDown-
load(
        $username,
        $privatekey,
        $publickey,
        'downloadFileList',
        NULL,
        $status,
        $startdate,
        $enddate
    );

    $client = new SoapClient(
        $this->wsdl,
        array(
            'trace' => TRUE,
            'exceptions' => FALSE
        )
    );

    $client->setPrivateKey($privatekey);
    $client->setPublicKey($publickey);

    $return = $client->__soapCall('downloadFileList', $arguments);

    if ($return->ResponseHeader->ResponseCode == '00') {
        $files = $this->CI->opapplicationresponse->parseFileList($re-
turn);
        return $files;
    } else {
        return FALSE;
    }
}
```

getFile-funktio BankWebService-luokassa

```
/**
 * @param $username
 * @param $privatekey
 * @param $publickey
 * @param $file
 *
 * Get specific file from Web service where filerefence (file_id)
matches
 * @return bool
 */
public function getFile($username, $privatekey, $publickey, $file) {

   $arguments = $this->CI->opapplicationrequest->getArgumentsToDown-
load(
      $username,
      $privatekey,
      $publickey,
      'downloadFile',
      $file);

   $client = new SoapClient(
      $this->wsdl,
      array(
         'trace' => TRUE,
         'exceptions' => FALSE
      )
   );

   $client->setPrivateKey($privatekey);
   $client->setPublicKey($publickey);

   $return = $client->__soapCall('downloadFile', $arguments);

   if ($return->ResponseHeader->ResponseCode == '00') {
      $file = $this->CI->opapplicationresponse->parseFile($return);
      return $file;
   } else {
      return FALSE;
   }
}
```

readReferencePaymentsToAccountsReceivable-funktio

```
/**
 * Reads reference payment material to accounts receivable.
 */
public function readReferencePaymentsToAccountsReceivable($str, $com-
pany_id = NULL, $isCron = FALSE) {

   try {
      $this->db->trans_start();

      $accountsbatch_id = $this->accountsreceivable_model->addAc-
countsBatch($company_id);

      $transaction_ids = array();

      foreach ($lines as $line) {
         $line = trim($line);

         if (mb_substr($line, 0, 1) === '3') {


            $accountseventtype_id = $this->accountsreceivable_model-
>getSystemReferencePaymentEventTypeId($company_id);

            $transaction = array(
               'accountseventtype_id' => $accountseventtype_id,
               'total' => $sum,
               'paydate_formatted' => $paydate,
               'valuedate_formatted' => $valuedate,
               'reference' => ltrim($reference, '0'),
               'name' => $shortname,
               'accountsbatch_id' => $accountsbatch_id,
               'sourceid' => trim($line)
            );

            $accountstransaction_id = $this->accountsreceivable_model-
>addTransaction(
               $transaction,
               $company_id
            );

            $transaction_ids[] = $accountstransaction_id;

            $imported++;
         }
      }}
}
```