Raine Pyssysalo

Laiterekisterien integrointi SeAMK:n ympäristössä - Biztalk Server 2006

Opinnäytetyö

Syksy 2009

Tekniikan yksikkö

Tietojenkäsittely

Sovellustuotanto

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖN TIIVISTELMÄ

Koulutusyksikkö: Tekniikan yksikkö
Koulutusohjelma: Tietojenkäsittelyn koulutusohjelma
Suuntautumisvaihtoehto: Sovellustuotannon suuntautumisvaihtoehto

Tekijä: Raine Pyssysalo

Työn nimi: Laiterekisterien integrointi SeAMK:n ympäristössä - Biztalk Server 2006

Ohjaaja: Markku Lahti

Vuosi: 2009 Sivumäärä: 104 Liitteiden lukumäärä: 9

Nykyisin sovelluksilta vaaditaan integraatiokykyä. Järjestelmät pitää saada keskustelemaan toistensa kanssa. Ei riitä, että uudet sovellukset on integroitu keskenään, vaan vanhat järjestelmät pitää saada liitettyä uusiin. Seinäjoen ammattikorkeakoulussa on monia erilaisia tietojärjestelmiä, jotka eivät vaihda tietoja keskenään. Tämä aiheuttaa monesti ongelmia, koska on erittäin työlästä hakea samaan aihepiiriin liittyvää tietoa monesta eri järjestelmästä. Microsoft vastaa integraatioongelmiin tuotteella, jonka nimi on Biztalk Server.

Tämän opinnäytetyön tavoitteena on antaa yksityiskohtainen kuvaus Seinäjoen ammattikorkeakoulun atk-henkilöstölle Biztalkin käytöstä yksinkertaisen esimerkkiprojektin avulla. Tässä projektissa yhdistetään kaksi eri tietolähdettä: Seinäjoen ammattikorkeakoulun oma SCCM-tietokanta ja kolmannen osapuolen Webservicerajapinnalla toimiva laiterekisteri. Lopuksi projekti julkaistaan Webservicenä ja sille luodaan asiakasohjelma. Opinnäytetyöstä hyötyvät myös muut sellaiset organisaatiot, jotka haluavat aloittaa Biztalk Server-projektin. Projektia voidaan pitää onnistuneena, koska alussa asetetut tavoitteet saavutettiin.

Asiasanat: Tietojärjestelmäintegraatio, Biztalk Server

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology
Degree programme: Business Information Technology
Specialisation: Application Production

Author: Raine Pyssysalo

Title of the thesis: Integration of device registers in Seinäjoki University of Applied Sciences - Biztalk Server 2006

Supervisor: Markku Lahti

Year: 2009 Number of pages:104 Number of appendices: 9

Nowadays there is demand for integrations between applications. The general opinion is that different systems should be able to talk to each other. It is not enough that new applications are integrated with each other, but old systems need to be integrated with new ones too. Seinäjoki University of Applied Sciences has many different systems that do not exchange data. This often causes problems because it is very laborious to fetch same kind of data from different systems. Microsoft has its own answer to this problem in the form of a product called Biztalk Server.

The aim of this thesis is to give the personnel of The School of Technology at Seinäjoki University of Applied Sciences a detailed description of how to use Biztalk with a simple example project. This project is about integrating two different data sources. These sources are the SCCM database of Seinäjoki University of Applied Sciences and a web service operated device register of a third party. Finally the project will be published as a web service, and a client program will also be created. This thesis gives a lot of knowledge also to other organizations which are planning to implement a Biztalk Server project. The project can be considered a success because all the goals that were set in the beginning of the project have been reached.

Keywords: data integration, Biztalk Server

SISÄLLYS

KÄYTETYT TERMIT JA LYHENTEET

C#	Microsoft-yhtiön .NET-arkkitehtuuria varten kehittämä ohjelmointikieli.
Asp.Net	ASP (Active Server Pages) on Microsoftin kehittämä dynaamisten www-sivujen luomiseen tarkoitettu palvelinpuolen ohjelmointimenetelmä. ASP.Net on ASP:n kehittyneempi versio, joka perustuu Microsoftin .NET-arkkitehtuuriin.
AssetWebservice	3stepIT-yrityksen tarjoama rajapinta laiterekisterihakuihin.
SCCM	System Center Configuration Manager on sovellus, joka keskittyy suurien laitteistokokonaisuuksien kokonaisvaltaiseen hallintaan.
Webservice	W3C:n määritelmän mukaan ohjelmistojärjestelmä, joka mahdollistaa keskenään yhteensopivan tietokoneiden välisen vuorovaikutuksen tietoverkon yli.
Microsoft Sql Server	Microsoftin kehittämä SQL-palvelintuote.
IIS	Internet Information Services on Microsoftin kehittämä webpalvelin.
SOAP	SOAP on protokolla, joka mahdollistaa Xml-muotoisen datan liikkuttaminen eri järjestelmien välillä. (Hunter, Watt, Rafter, Duckett, Ayers, Chase, Fawcet, Gaven, Patterson 2004, 559.)

Kuviot

Taulukot

1 JOHDANTO

Nykyisin sovelluksilta vaaditaan integraatiokykyä. Järjestelmät pitää saada keskustelemaan toistensa kanssa. Ei riitä, että uudet sovellukset on integroitu keskenään, vaan vanhat järjestelmät pitää saada liitettyä uusiin. Seinäjoen ammattikorkeakoulussa on monia erilaisia tietojärjestelmiä, jotka eivät vaihda tietoja keskenään. Tämä aiheuttaa monesti ongelmia, koska on erittäin työlästä hakea samaan aihepiiriin liittyvää tietoa monesta eri järjestelmästä. Microsoft vastaa integraatioongelmiin tuotteella, jonka nimi on Biztalk Server.

Tämän opinnäytetyön tavoitteena on antaa yksityiskohtainen kuvaus Seinäjoen ammattikorkeakoulun atk-henkilöstölle Biztalkin käytöstä yksinkertaisen esimerkkiprojektin avulla. Tässä projektissa yhdistetään kaksi eri tietolähdettä: Seinäjoen ammattikorkeakoulun oma SCCM-tietokanta ja kolmannen osapuolen Webservice-rajapinnalla toimiva laiterekisteri. Opinnäytetyöstä hyötyvät myös muut sellaiset organisaatiot, jotka haluavat aloittaa Biztalk Server -projektin.

Biztalk Server -tuotetta voidaan pitää yhtenä nykypäivän parhaiten markkinoituna integraatiotuotteena. Biztalk nimi mainitaan usein tietotekniikka-alalla, silloin kun puhutaan integraatioprojekteista. Vapaanlähdekoodin parissa etsitään jatkuvasti korvaavaa järjestelmää Biztalk Serverille. Kaikki nämä järjestelmät perustuvat Xml-muotoisiin viesteihin ja Webservice-tekniikoihin. Biztalkin huonona puolena voidaan pitää sen hintaa ja sitä, että osaajia on hyvin vähän. Biztalk-projektin hinta tulee olemaan korkea, jos pitää ostaa palvelin, tuote ja vielä lopuksi palkata projektille osaaja.

1.1 Työn tausta

Seinäjoen ammattikorkeakoululla on Biztalk Server, jota olisi mahdollisuus käyttää järjestelmien integrointiprojekteihin. Biztalkista on kuitenkin hyvin vähän käytännön projektiesimerkkejä. Tällä opinnäytetyöllä yritetään vastata tähän tarpeeseen.

Seinäjoen ammattikorkeakoululla on käytössään Microsoftin tuote SCCM (System Center Configuration Manager). Tätä järjestelmää käytetään laitteiden hallintaan. Kaikkiin laitteisiin asennetaan esimerkiksi päivitykset keskitytetysti tämän sovelluksen avulla. Tuote käyttää Microsoft Sql Server -tietokantaa. Tätä tietokantaa on käytetty hyväksi tässä opinnäytetyössä. (Ilenius 2009, 1.)

Seinäjoen ammattikorkeakoulu on 3stepIT-nimisen yrityksen asiakas, joka toimittaa sille erilaisia laitteita. 3stepIT:llä on olemassa palvelu, jolla asiakas voi hakea omien laitteidensa tietoja Xml-muotoisena. Tällä palvelulla voidaan hakea esimerkiksi tietoja erilaisista sopimuksista ja laitteiden hinnoista.

1.2 Työn tavoitteet

Opinnäytetyön tavoite on saada kaksi eri tietolähdettä integroitua toisiinsa. Molemmat tietolähteet sisältävät tietoja erilaisista laitteista. Integroinnissa käytetään yhdistävänä tekijänä laitteen sarjanumeroa. Integroitavat tietolähteet ovat SCCM-tietokanta ja AssetWebService. Integroinnilla tavoitellaan myös mahdollista jatkokehitystä.

Nykyisin Seinäjoen ammattikorkeakoulussa käytetään kumpaankin tietolähteeseen omaa erillistä sovellusta. Tavoitteena on toteuttaa käyttöliittymä, josta voi hakea tietoa syöttämällä laitteen sarjanumeron hakukenttään. Käyttöliittymä suunnitellaan web-sivuksi.

1.3 Työn rakenne

Opinnäytetyön toisessa luvussa esitellään Seinäjoen ammattikorkeakoulun ympäristö. Luvussa kuvataan eri järjestelmät, jotka ovat olleet mukana kehitystyössä. Luvussa käydään myös läpi integraatiota, joita Seinäjoen Ammattikorkeakoululla on jo olemassa.

Kolmannessa luvussa käsitellään Biztalk-järjestelmän perusteita. Luvussa edetään perustiedoista yksityiskohtaisempiin määritelmiin. Perusteisiin kuuluvat esimerkiksi Xml-dokumentit ja Xml Schemat. Yksityiskohtaisemmista määritelmistä voidaan mainita sarjallistusluokat. Xml-kuvauskieleen liittyviä asioita käydään läpi, koska niiden ymmärtämisen jälkeen on helpompi perehtyä Biztalk-palvelimeen.

Neljännessä luvussa käsitellään Biztalk-palvelimen arkkitehtuuri. Luvun alussa kuvataan Biztalkin arkkitehtuuria kokonaisuutena. Tämän jälkeen esitellään Biztalkin kaikki tärkeimmät komponentit ja niiden toiminnallisuus. Tärkeimpiin komponentteihin kuuluvat esimerkiksi viestit ja orkestraatiot.

Viidennessä luvussa siirrytään käytännön tasolle Biztalk esimerkkiprojektin avulla. Luvussa käydään aluksi läpi molempien tietolähteiden arkkitehtuuri. Sen jälkeen siirrytään itse projektin luomiseen ja sen rakentamiseen, sekä käydään läpi tärkeimpien komponenttien konfiguroinnit. Luvun lopussa esitellään Webservice, joka projektin tuloksena syntyi.

Kuudennessa luvussa kerrotaan sovelluksen mahdollisesta jatkokehityksestä. Luvussa käydään myös läpi mihin Biztalk-järjestelmää kannattaa käyttää. Luvussa kerrotaan myös, miksi järjestelmää ei aina kannata käyttää jokaiseen tietojenkäsittelyongelmaan.

Seitsemännessä luvussa tehdään yleisiä johtopäätöksiä työn onnistumisesta.

2 SEAMK:N LAITEREKISTERIT JA NIIDEN HALLINTA

Seinäjoen ammattikorkeakoulussa on käytössä SCCM (System Center Configuration Manager). Tämä sovellus on tarkoitettu isojen organisaatioiden laitteiden hallintaan. Sovellusta käytetään myös tietojärjestelmien optimointiin, sekä erilaisten raporttien luomiseen eri laitteistokokonaisuuksien tilasta. Sovellus käyttää SCCM-tietokantaa, jossa on tietoja laitteista. SCCM toimii siis samalla laiterekisterinä. SCCM on ensimmäinen järjestelmistä, josta tietoja halutaan saada. (System Center Configuration Manager: Overview. 2009)

Seinäjoen ammattikorkeakoulu on 3stepIT-nimisen yrityksen asiakas. Yritys tarjoaa palveluna AssetWebService-nimisen rajapinnan. AssetWebserviceä on tarkoitus käyttää Biztalkista suoraan ja yhdistää tiedot SCCM-tietokannan kanssa.

Seinäjoen ammattikorkeakoulun integraatioista voidaan mainita metahakemistojen käyttö käyttäjien autentikoinnissa eri järjestelmiin. Käyttäjät kykenevät metahakemiston ansiosta kirjautumaan samalla tunnuksella ja salasanalla eri järjestelmiin. Integraatioiden tarkoitus on sulauttaa eri järjestelmät yhdeksi suuremmaksi kokonaisuudeksi. Metahakemistot Seinäjoen ammattikorkeakoulussa toteuttavat tätä ajatusta hyvin.

Integraatiomahdollisuudet Biztalk Server näkökulmasta ovat hyvät Seinäjoen ammattikorkeakoulussa. Biztalk voidaan asentaa mille tahansa Windows-palvelimelle. Opinnäytetyön kehitystyön aikana käytettiin virtuaalipalvelinta, johon otettiin aina yhteys etätyöpöytäyhteyden avulla. Kehityspalvelin oli suorituskyvyltään heikompi, kuin tuotantokäytössä oleva. Palvelimella oli kehitystyökaluna Microsoftin Visual Studio 2005. Tietokantapalvelimena toimi Microsoft Sql Server 2005. Palvelimelle oli myös asennettu IIS (Internet Information Services) web-palvelin. Kehittäjän näkökulmasta palvelimen suorituskyky riitti hyvin kehitystyöhön.

3 XML-KIELI BIZTALKIN TAUSTALLA

Xml muotoinen data muodostaa Biztalkin viestijärjestelmän perustan. Ilman tuntemusta Xml:n eri osa-alueista on vaikea lähteä rakentamaan Biztalkissa toimivaa sovellusta. Tässä luvussa käydään läpi viisi eri XML:n osa-aluetta.

3.1 Xml-dokumentti

Xml-dokumentti muodostuu laajennettavasta merkintäkielestä (eXtensible Markup Language.) Xml on metamerkintäkieli, jossa voi itse luoda tarvitsemansa merkintäkoodit. Merkintäkoodeja kutsutaan elementeiksi. Elementti voi myös sisältää attribuutin, joka kuvaa tiettyä elementtiä paremmin. Xml kuvaa dokumentin rakenteen muttei muotoilua. Muotoiluun voidaan erikseen käyttää esimerkiksi Css-tyylitiedostoa. Tämän vastakohtana on HTML, joka sisältää valmiiksi muotoiluun ja rakenteeseen liittyvät merkintäkoodit. (Harold 2000, 28.)

Esimerkissä 1 on Xml-dokumentti, jolla voitaisiin viedä Biztalkin läpi viesti, joka tekisi tilauksen tietystä kirjasta. Xml-dokumentti alkaa prologilla, joka sisältää dokumentin koodauksen ja Xml:n version. Koodaus on tärkeä osa Biztalkin läpi kulkevaa viestiä. Jos koodaus on eri kuin vastaanottavassa järjestelmässä, niin esimerkiksi skandinaaviset kirjaimet voisivat näkyä väärin.

Prologin alla on erilaisia elementtejä, jotka kuvaavat varsinaisen tiedon. Tieto tulee aina elementin aloitus- ja lopetusmerkinnän väliin. Tietotyyppejä voidaan määritellä Xml:n rakenteen kuvaavassa dokumentissa kuten Xml Schema. Xml:n on tärkeää olla hyvin muodostettua. Esimerkiksi Biztalkissa viestin toimittaminen kohdejärjestelmään voisi keskeytyä Xml-dokumentin puutteiden vuoksi. Kyseessä voisi olla vaikka yksinkertainen kirjoitusvirhe elementissä.

Esimerkki 1. Xml-dokumentti

```
<?xml version="1.0" encoding="UTF-8"?>
<kirjakauppa>
  <kirja>
    <kirjoittaja></kirjoittaja>
    <nimi></nimi>
    <julkaisija></julkaisija>
    <vuosi></vuosi>
    <tyylilaji></tyylilaji>
  </kirja>
</kirjakauppa>
```

3.2 Xml Schema

Schema on kieli, jolla kuvataan Xml-dokumentin rakenne, säännöt ja rajoitukset. Monessa tapauksessa Schema on kehittäjälle suunnittelutyökalu. Se toimii perustana, jolla voidaan rakentaa sovellus. Scheman tärkein tehtävä on Xml:n tarkistaminen. Tarkistaminen toimii monessa järjestelmässä palomuurina. Näin varmistetaan, että ulkopuolelta tulevassa Xml-muotoisessa datassa ei ole virheitä. Virheelliset viestit voisivat tehdä tuhoa vastaanottavassa järjestelmässä, jollei niitä tarkistettaisi ensin. (Vlist 2002, 13.)

Scheman on kehittänyt W3C (The World Wide Web Consortium). Alussa oli olemassa vain DTD (Document Type Definition) Xml-dokumentin rakenteen määrittelyyn. Kehittäjät huomasivat kuitenkin kaipaavansa enemmän monipuolisuutta määrittelyyn. Schema tarjoaa paljon monipuolisemmat mahdollisuudet esitellä esimerkiksi tietotyyppejä DTD-malliin verrattuna. (Vlist 2002, 16.)

Esimerkissä 2 kuvattu Schema määrittelee elementin nimeltä Henkilo, joka on ComplexType. Tämä tarkoittaa, että elementillä voi olla kokoelma alielementtejä. Scheman jälkeen on kuvattu Scheman generoima Xml-dokumentti. Nyt tätä Schemaa voidaan käyttää sääntönä, jota pitää soveltaa esimerkiksi kehitetyssä palvelussa. (Jefford, Smith, Fairweather 2007, 2.)

Esimerkki 2. Xml Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Henkilo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Nimi" type="xs:string"/>
        <xs:element name="Ika" type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Vastaava Xml-dokumentti

```
<?xml version="1.0" encoding="UTF-8"?>
<Henkilo>
  <Nimi></Nimi>
  <Ika></Ika>
</Henkilo>
```

Schemat ovat tärkeä osa Biztalkin viestijärjestelmää. Niiden avulla Biztalk kykenee tekemään erilaisia operaatioita ja tarkistamaan, että sisään tulevassa viestissä ei ole virheitä. Schemat ovat myös erittäin tärkeitä, kun Biztalkissa hyödynnetään Webservice-rajapintaa. Webservice-instanssien kanssa kannattaa olla tarkkana, jos konvertoi DTD-tiedoston Schemaksi tai toisinpäin. Useimmiten nämä kaksi tiedostoa eivät vastaa toisiaan. Nykyaikana Webservicet pitäisi aina määritellä Scheman avulla.

3.3 Xml-nimiavaruudet

Nimiavaruuksia käytetään usein Xml-dokumenteissa ja ne ovat Biztalkin viestijärjestelmän tärkeä osa. Nimiavaruuden tarkoitus on pitää Xml-dokumentin elementit yksilöivinä. Nimiavaruus pitää huolen siitä, että jokaisella viestillä on oma Schemansa. Nimiavaruudet voivat aiheuttaa ongelmia järjestelmäintegraatioissa, kun viestin nimiavaruus ei vastaa Scheman nimiavaruutta. Nimiavaruus määritellään Xml-dokumentin prologissa: xmlns:”http://Henkilo”. Tämä nimiavaruus voisi kuulua

edellä esitellylle Xml-dokumentille ja Schemalle. (Jefford, Smith, Fairweather 2007, 3.)

Usein oletetaan, että nimiavaruudessa käytetyn URL-osoitteen pitää osoittaa johonkin konkreettiseen kohteeseen esimerkiksi web-palvelimella. Osoite voi olla mikä tahansa keksitty osoite, joka on tarpeeksi yksilöivä kyseessä olevalle Xml-dokumentille. (Jefford, Smith, Fairweather 2007, 3.)

3.4 Xpath

Xpath on navigointikieli, jolla voidaan liikkua Xml-dokumentilla. Sen tärkein tehtävä on löytää tietoja erilaisten ehtojen avulla. Xpath-haku tehdään lausekkeella (regular expressions). Biztalkissa viesteistä voidaan hakea tiettyjä tietoja, joiden avulla tiedot voidaan reitittää oikeaan paikkaan tai asettaa niille ehtoja. Xpath-lauseke käyttää samaa syntaksia, kuin Windowsin tiedostojärjestelmä. Xpath-tietämyksen avulla on mahdollista ratkaista haastavia ongelmia Biztalkissa. (Jefford, Smith, Fairweather 2007, 3.)

Tässä opinnäytetyössä ei käytetty Xpathia Biztalkin sisällä. Sitä käytettiin myöhemmin C#-ohjelmassa, joka tekee tietokantapäivityksen öisin. Ohjelma hakee Webservicen avulla Xml-virran, josta pitää lukea tarvittavat tiedot ehtojen avulla. Kyseisessä ohjelmassa Xpath-funktio on seuraavanlainen:

```
oneObject.SelectSingleNode("attribute[@code='name']").InnerText)
```

Tässä funktiossa Xpath hakee ehdon avulla elementin sisältämän tekstin, joissa attribuutin ”code” arvo on ”name”.

Xpathiin on sisäänrakennettu monia hyödyllisiä funktioita, kuten count-funktio. Tätä funktiota kannattaa käyttää, kun halutaan tietää kuinka monta kertaa tietty objekti esiintyy Xml-dokumentilla. Esimerkiksi tietokannan insert-lausekkeen suorit-

tamisessa saatetaan tarvita luuppia. Tällöin objektien esiintymismäärä voitaisiin antaa ehdoksi luupin lopettamiselle. (Jefford, Smith, Fairweather 2007, 5.)

3.5 Xml-serialisointi

Xml serialisointiluokat ovat tärkeitä Biztalkin ulkopuolella toimiville asiakasohjelmille. Esimerkiksi silloin kun on kyseessä SOAP-kutsu, niin luokat hoitavat Xml-dokumenttien käsittelyn. Luokan avulla SOAP-kutsulle asetetaan parametri. Luokka vastaanottaa vastauksen, joka vastaa .Net-ympäristössä oliota eli objektia. Objektista on helppo hakea tarvittavat tiedot metodilla, jolla luetaan attribuutin tiedot. (Jefford, Smith, Fairweather 2007, 8.)

3.6 Serialisointiluokan generointi Schemasta

Xml-serialisointiluokkia voidaan generoida Visual Studion mukana tulevalla ohjelmalla. Ohjelma löytyy Visual Studion komentokehotteesta. Ohjelman voi ajaa hakemiston juuressa. Xml-serialisointiluokka generoidaan komennolla: ”XSD.exe /C henkilo.xsd”. Generoinnissa käytetään Esimerkin 2 Schemaa. Ohjelma generoi henkilo.cs-nimisen luokan samaan hakemistoon. Esimerkissä 3 esitetään tuloksena generoitunut Xml serialisointi-luokka. (Jefford, Smith, Fairweather 2007, 9.)

Esimerkki 3. Xml-serialisointiluokka

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3053
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System.Xml.Serialization;
```

```
//
// This source code was auto-generated by xsd, Version=2.0.50727.42.
//


/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "2.0.50727.42")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType=true)]
[System.Xml.Serialization.XmlRootAttribute(Namespace="",
IsNullable=false)]
public partial class Henkilo {

    private string nimiField;

    private int ikaField;

    /// <remarks/>

[System.Xml.Serialization.XmlElementAttribute(Form=System.Xml.Schema.XmlS
chemaForm.Unqualified)]
    public string Nimi {
        get {
            return this.nimiField;
        }
        set {
            this.nimiField = value;
        }
    }

    /// <remarks/>

[System.Xml.Serialization.XmlElementAttribute(Form=System.Xml.Schema.XmlS
chemaForm.Unqualified)]
    public int Ika {
        get {
            return this.ikaField;
        }
        set {
            this.ikaField = value;
        }
    }


}
```

Yllä kuvattu luokka voisi toimia vaikka rekisteröitymispalveluna, jolle välitetään parametrina vain nimi ja ikä. Luokasta puuttuu vielä tarvittava palvelun kutsumetodi, mutta tällä luokalla voisi jo muodostaa Xml-dokumentin. Luokka varmistaa myös Xml-dokumentin oikeellisuuden. Varmistuksessa voisi tulla ongelmia ainoastaan siinä tapauksessa, jos luokan generoija on tehnyt virheen Scheman määrittelyssä.

Useimmiten kuitenkin kehittäjä saa Schemat valmiina, jos hänen pitää kutsua jotakin palvelua. Tällöin vikavastuu siirtyy toiselle osapuolelle.

Xml-serialisointiluokat ovat tehokkaita silloin, kun ei käytetä liian suuria viestejä. Tuhansia rivejä sisältävään viestiin ei kannata käyttää näitä luokkia, koska kaikki tieto ladataan muistiin. Tämä voi kasvattaa turhan prosessoritehon käyttöä. Serialisointiluokat kuitenkin käyttävät samaa AppDomainia kuin Biztalk, joka toimii eräänlaisena välimuistina. (Jefford, Smith, Fairweather 2007, 11.)

4 BIZTALK-ARKKITEHTUURI

On tärkeää muistaa, että Biztalk ei ole universaali ratkaisu kaikkiin ohjelmistokehityksen ongelmiin. Biztalkin käyttöä on harkittava tarkasti. Biztalk Serveriä tulisi käyttää vain silloin, kun se on järkevää. Muutoin Biztalk voi muodostua pullonkaulaksi liiketoimintaprosessissa. (Jefford, Smith, Fairweather 2007, 15.)

Kuviossa 1 on kuvattu Biztalkin arkkitehtuuri ja viestin liikkuminen eri komponenttien läpi. Näitä komponentteja ovat: viesti (Message), vastaanottava portti (Receive Port), vastaanottopaikka (Receive Location), kartat (Maps), viestilaatikko (Messagebox), orkestraatio (Orchestration) ja lähettävä portti (Send Port). (Dunphy, Metwally 2006, 67.)

Viesti: Viesti on tieto, joka liikkuu Biztalkin läpi. Tieto esitetään viestinä riippumatta sen sisällöstä. Viesti toimii Biztalkin toiminnallisuuden perustana. (Dunphy, Metwally 2006, 68.)

Vastaanottava Portti: Sisääntuleva viesti kulkee aina vastaanottavan portin kautta Biztalkissa. Vastaanottava portti mahdollistaa viestin siirtämisen orkestraation käyttöön. (Dunphy, Metwally 2006, 68.)

Vastaanottopaikka: Vastaanottavalle portille määritellään aina vastaanottopaikka. Tämä paikka voi olla esimerkiksi kansio tiedostojärjestelmässä tai http-URL. (Dunphy, Metwally 2006, 68.)

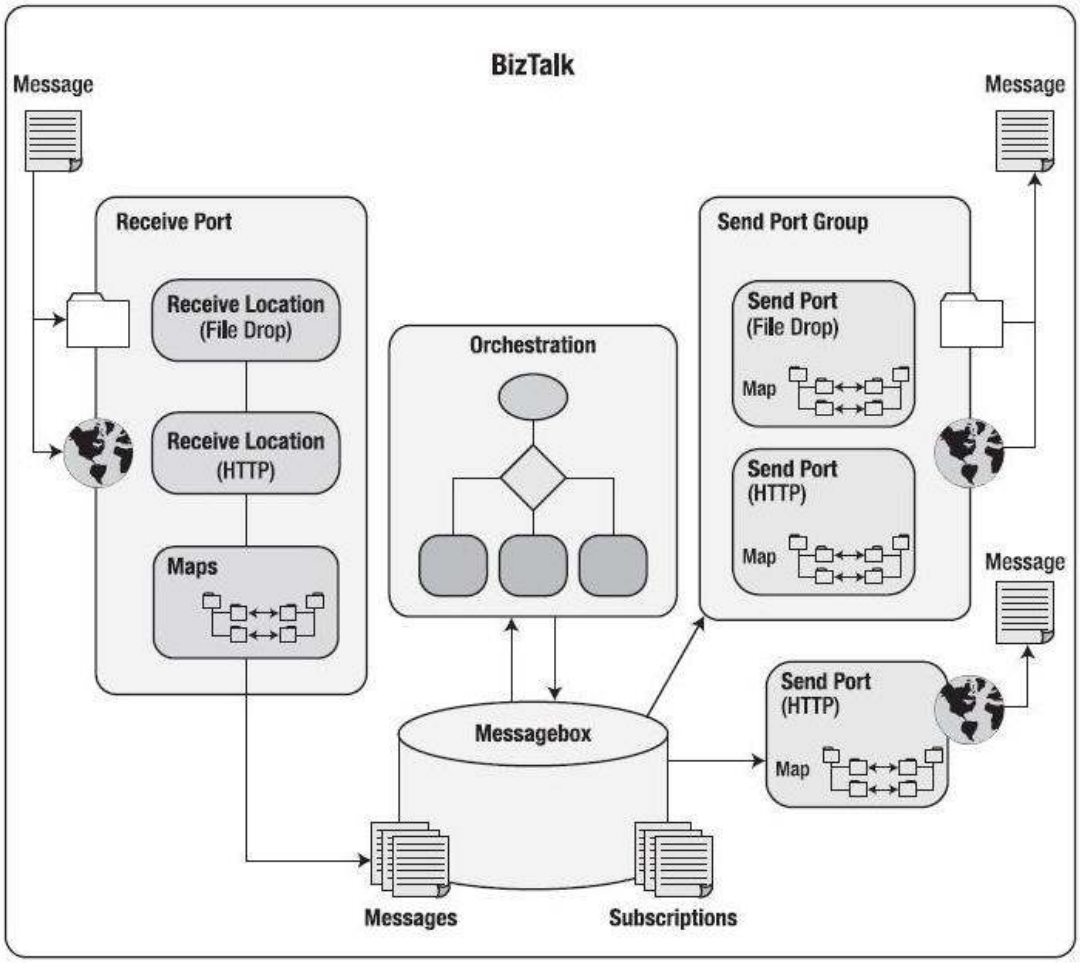
Kartoitus: Kartoitukset muokkaavat viestin haluttuun muotoon. Tämän konvertoinnin jälkeen viesti voidaan viedä esimerkiksi johonkin toiseen järjestelmään. (Dunphy, Metwally 2006, 68.)

Viestilaatikko: Viestilaatikko on tietokanta, johon tallennetaan monenlaisia tietoja viesteistä. (Dunphy, Metwally 2006, 68.)

Orkestraatio: Orkestraatio kuvaa sovelluksen prosessin ilman mittavaa määrää lähdekoodia. Orkestraatiossa esitetään sovelluksen toimintalogiikka graafisesti. (Dunphy, Metwally 2006, 68.)

Lähettävä portti: Lähettävä portti lähettaa viestin järjestelmän ulkopuolelle. (Dunphy, Metwally 2006, 68.)

Pipeline: (Putki) Putket hoitavat erilaisissa formaateissa olevien viestien prosessoinnin sekä toteuttavat eri protokollia. (Dunphy, Metwally 2006, 68.)



Kuvio 1. Biztalk arkkitehtuuri (Dunphy, Metwally 2006, 68.)

Biztalkin arkkitehtuuri perustuu ns. viestilaatikko (MessageBox) ajatteluun. Viestit ovat useimmiten Xml-muotoisia ja perustuvat johonkin Schemaan. Viesti voi olla

myös binääri-muotoista dataa. Viesti saapuu viestilaatikkoon vastaanottopaikan kautta eli fyysisen portin kautta. Tästä viesti jatkaa putken kautta viestilaatikkoon, josta se siirretään orkestraatioon. Seuraavana viesti jatkaa adapterille. Viestin saavuttua viestilaatikkoon sitä ei voi enää muuttaa. (Dunphy, Metwally 2006, 67.)

Viestilaatikko on yksinkertaisesti tietokanta. Tässä tietokannassa on useita tauluja, joihin tallennetaan viestit ja niiden tiedot. Tietokantaan tallennetaan myös metadataa, jossa on esimerkiksi seuraavia tietoja viestistä:

- viestin tuloportti
- viestin välitystyyppi
- tiedoston nimi
- viestin ID-tunniste
- Scheman tyyppi ja nimiavaruus. (Dunphy, Metwally 2006, 69.)

4.1 Viestilaatikon komponentit

Viestilaatikossa on Host service, joka on looginen säilytyspaikka viesteille. Host service jakaa viestien osat eri osapuolille Biztalk-projektissa. Se jakaa orkestraatiot, portit ja adapterit omiin ryhmiinsä. Host Servicen tarkoitus on siis jakaa tehtävät eri instansseille jakaakseen muistikuormaa. Host Service ylläpitää kahta eri Host-tyyppiä Isolated Host ja In-Process Host. Näiden ero on se, että Isolated Host suoritetaan ulkopuolisen palvelun alaisuudessa. Useimmiten tämä palvelu on IIS. In-Process Host alaisuudessa suoritetaan pelkästään Biztalk projektiin kuuluvia osioita. (Dunphy, Metwally 2006, 69.)

Subscriptions eli tilaukset Biztalkissa ovat mekanismi, jonka avulla Biztalk osaa vastaanottaa ja lähettää viestejä orkestraatioiden tai porttien läpi. Microsoftin mukaan tilaus on kokoelma tunnisteita, joiden avulla viesti pystytään ohjaamaan oikeaan paikkaan. (Dunphy, Metwally 2006, 70.) Tilaukset aiheuttivat ongelmia tässä projektissa. Tilaukset määrittävät Biztalkin tietokantaan myös nimiavaruudet.

Jos viestin nimiavaruus ei vastaa tilausta, niin viestin välitys epäonnistuu. Kyseessä oli pieni kirjoitusvirhe. Virheen selvittämiseen meni kuitenkin aikaa, koska Biztalk hallintakonsolin virheilmoitukset eivät aina ole niin selkeitä kun saattaisi toivoa.

Enlisting on Biztalkissa porttien tai orkestraation aktivoimista. Aktivoinnissa kyseisen instanssin tiedot kirjoitetaan Biztalkin tietokantaan. Aktivoiminen ei esimerkiksi tarkoita portin käynnistämistä tuotantokäyttöön. Aktivoinnissa kirjoitetaan vain tarvittavat attribuutit tietokantaan. (Dunphy, Metwally 2006, 71.)

4.2 Viestit

Viestit ovat jokaisen Biztalk-sovelluksen perusta. Useimmiten viestit ovat Xml-dokumentteja, mutta ne voivat joskus myös olla binäärisiä. Binäärisien viestien muuttaminen ei käy yhtä helposti kuten Xml-viestien. Viestin tärkein osa on Body-osa, joka sisältää dataa. Jos käyttäjä haluaa jakaa viestin datan useampaan osaan, niin silloin voidaan käyttää Multipart-viestiä. Biztalk tarjoaa myös työkalun seurata viestien kulkua järjestelmässä. HAT (Health and Activity Tracking) -työkalulla voi selvittää viestin kulkua etenkin ongelmatapauksissa. HAT näyttää tarkasti missä viestin kulku päättyy. (Jefford, Smith, Fairweather 2007, 21.)

Orkestraatioissa viestit sidotaan orkestraation elementteihin. Yleensä Biztalk orkestraatiota testataan testiviestillä, joka laitetaan kiertoon tiedostojärjestelmän kautta. Testiviesti on helppo generoida Schemasta, joka määrittelee viestin. Scheman ominaisuuksista voi määritellä tiedoston luontipaikan. Tiedoston voi generoida Generate Instance -valinnalla. (Jefford, Smith, Fairweather 2007, 57.)

4.3 Adapterit

Biztalkissa adapterit ovat ovi ulkopuoliseen maailmaan. Adapterit hoitavat liikenteen Biztalkkiin, sekä sieltä kohdejärjestelmään. Ne toteuttavat tarvittua kommuni-

kaatiokaavaa (Communication Pattern) tai protokollaa, jota ulkopuolinen järjestelmä vaatii. Ne piilottavat myös etäjärjestelmien monimutkaisuuden. Adaptereita pidetään ”bitinsyöttäjinä”, koska ne syöttävät Biztalkiin bittejä, joita se työstää ja lähettää takaisin ulkopuoliseen maailmaan. Adapterit eivät suorita minkäänlaisia konvertointioperaatioita viesteille. Ne vain tuovat viestit järjestelmään tai lähettävät ne ulkopuolelle. Pipelinet hoitavat konversiot. Esimerkiksi tekstitiedoston muuntaminen Xml-muotoon voisi olla tällainen konversio. (Jefford, Smith, Fairweather 2007, 37.)

Adapterit ovat Biztalk järjestelmän lisäosia. Biztalk Server 2006 -versiossa on monia adaptereita jo valmiiksi sisäänrakennettuna. Kolmannen osapuolen toimijat tarjoavat myös lisää adaptereita maksua vastaan. Vuoden 2006 versioon on sisällytetty paljon uusia adaptereita asiakkaiden pyynnöstä, Microsoft on kehittänyt ohjelmointirajapinnan, jolla käyttäjä voi itse kehittää omia adaptereitaan. (Jefford, Smith, Fairweather 2007, 37.)

Biztalk-adapterit tukevat yksisuuntaista ja kaksisuuntaista viestien kulkukaavaa (Communication Pattern), kun viestit tulevat tai lähtevät järjestelmästä. Adapteri voi tukea molempia tekniikoita. Kaksisuuntaisia kulkukaavoja kutsutaan nimellä ”request-response”. Yksisuuntaisia kulkukaavoja kutsutaan nimellä ”solicit-response”. File-adapteri tukee vain yksisuuntaista kulkukaavaa. File-adapterille ei pysty määrittämään palautuspolkua. SOAP-adapteri käyttää tyypillisesti kaksisuuntaista kulkukaavaa. Sen voi myös määrittää yksisuuntaiseksi, ja tällöin se ei odota vastausta. SOAP-adapteri tukee molempia kulkukaavoja. (Jefford, Smith, Fairweather 2007, 39.)

Biztalk sisältää valmiiksi seuraavat adapterit: File, MSMQT, MSQM, Windows Sharepoint Services, SQL, SOAP, http, SMTP, POP3, EDI, MQSeries, FTP, WCF ja WCE. Tässä opinnäytetyössä on käytetty File, WCFcustom ja SOAP-adapteria. File-adapteria on hyvä käyttää projektin yleiseen testaukseen. Opinnäytetyössä käytettiin myös alun perin SQL-adapteria, mutta se huomattiin liian monimutkaiseksi käyttää. WCFcustom-adapteri on uusimpia lisäyksiä Biztalkin adapteriper-

heeseen. WCFcustom-adapteri on kehittyneempi kuin edellinen SQL-adapteri. Vanhassa adapterissa tietokantaproseduurit piti muuttaa ensin Xml-muotoon ennen kuin niitä pystyi käyttämään. SOAP-adapteri oli luonnollinen valinta Webservice-kutsuja suunnitellessa. SOAP-adapteri on huomattavasti nykyaikaisempi, kuin esimerkiksi http-adapteri.

4.3.1 File-adapteri

File-adapteri on useimmiten käytetty adapteri ja myös suosituin. Tämä adapteri toimii tiedostojärjestelmän kautta. Biztalk odottaa valmiiksi määriteltyyn hakemistoon saapuvaa tiedostoa, jolla on tietyntyyppinen tiedostopääte. Useimmiten tämä tiedostopääte on ”.xml”. Päätteen voi myös määrittää joksikin muuksi. Esimerkiksi kun tiedostoa muutetaan Xml-muotoon. File-adapteri toimii kahdessa eri tilassa. Adapterin huomatessa uuden tiedoston se lukitsee tiedoston omaan käyttöönsä. Adapteri pitää tiedoston lukittuna, kunnes se on läpäissyt pipelinen ja se on kirjoitettu viestilaatikkoon. Jos viestinkulku onnistuu orkestraation läpi, niin tiedosto tuhotaan tiedostojärjestelmästä. Jos viestinkulku epäonnistuu, niin tiedosto palautetaan kiertoon. (Jefford, Smith, Fairweather 2007, 39.)

Tiedoston jäädessä kiertoon kannattaa tarkistaa Administration Consolesta kyseessä oleva virhe. Konsolista voi myös käynnistää uudelleen kierron, jos kyseessä oleva virhe on tullut korjattua. Konsolista näkee myös tarkasti, minkä elementin kohdalla virhe on tapahtunut. Useimmiten kannattaa kuitenkin tuhota kaikki kierrot ja käynnistää Host Instance uudelleen virheiden jälkeen.

4.3.2 SOAP-adapteri

SOAP-adapteri mahdollistaa Biztalkissa viestien lähetyksen ja vastaanoton Webservicen kautta, joka on SOAP-protokollan mukainen ja tukee http-välitystä. Biztalkilla ei ole kykyä käyttää http-portteja ulkopuoliseen maailmaan menevälle liiken-

teelle. Sen sijaan se käyttää ASP.Netin Webservicejä, joita ylläpidetään IIS-palvelimella. Adapteri on ”eristetty”, joka tarkoittaa sitä että adapteri ylläpitää Biztalkin viestilaatikkoa eikä toisinpäin. SOAP-adapteri käyttää .Net-puolen määrityksiä Webserviceistä, siksi se antaa ottaa korkeintaan kaksi eri yhteyttä eri web-palvelimiin. Muut Webservicekutsut jätetään pitoon siihen asti kunnes ensimmäiset kutsut on suoritettu. Tämä ei ole huono asia, vaikka niin voitaisiin olettaa, koska se nojaa http 1.1 -standardiin, jonka mukaan yhden asiakasohjelman ei saisi ylläpitää enempää kuin kahta yhteyttä minkään palvelimen kanssa. Webservicet toteuttavat yleensä pyyntö-vastaus-periaatetta. SOAP-adapteri pystyy myös pelkkään pyyntöön, mutta silloin ei ole mitään varmuutta onnistuiko kutsu ja saatiinko tarvittavat muutokset tehtyä. Tärkeintä SOAP-adapterin käytössä olisi vastaanottaa vähintään http-koodi ”200”, joka tarkoittaa, että vastaanottava järjestelmä pystyi lukemaan viestin oikein. Kriittisissä ympäristöissä koodi ”200” ei välttämättä aina riitä takuuksi transaktion onnistumiselle. Esimerkiksi sähkökatkon aikana järjestelmä voi palauttaa onnistuneen koodin, vaikka jokin asia olisikin jäänyt tekemättä. (Jefford, Smith, Fairweather 2007, 73.)

SOAP-adapterin tärkeänä apuna ovat serialisointi-luokat. Ne auttavat saamaan lähetettävät ja vastaanotettavat tiedot paikoilleen. Tiedot pysyvät oikeellisina ja näin voidaan varmistaa, että vastaanottavassa Webservicessä ei tule virheitä. Tämä pätee yleensä silloin, kun käytetään asiakasohjelmaa, jolla kutsutaan Biztalkin generoimaa Webserviceä. Silloin kun kutsutaan Webserviceä Biztalkista käsin, niin Biztalk generoi Schemat valmiiksi tiedon lähettämiseen ja vastaanottamiseen. (How to Use the BizTalk Web Services Publishing Wizard to Publish an Orchestration as a Web Service. 2009.)

4.3.3 WCFcustom-adapteri

WCFcustom-adapteri (Windows Communication Foundation)-adapteri tuli versioon Biztalk Server 2006 R2 mukana sisäisiin adaptereihin, joita ei tarvitse erikseen asentaa. Tämän opinnäytetyön aikaan adapteri piti kuitenkin erikseen asentaa,

koska Biztalkin versio oli Biztalk Server 2006. Adapteri piti määrittää käyttämään tiettyä sidontaa. Tässä tapauksessa se oli luonnollisesti Sql-sidonta. Muita sidontatapoja ovat mm. OracleDbb, OracleEbs, Sap sekä Siebel.

4.4 Pipelinet

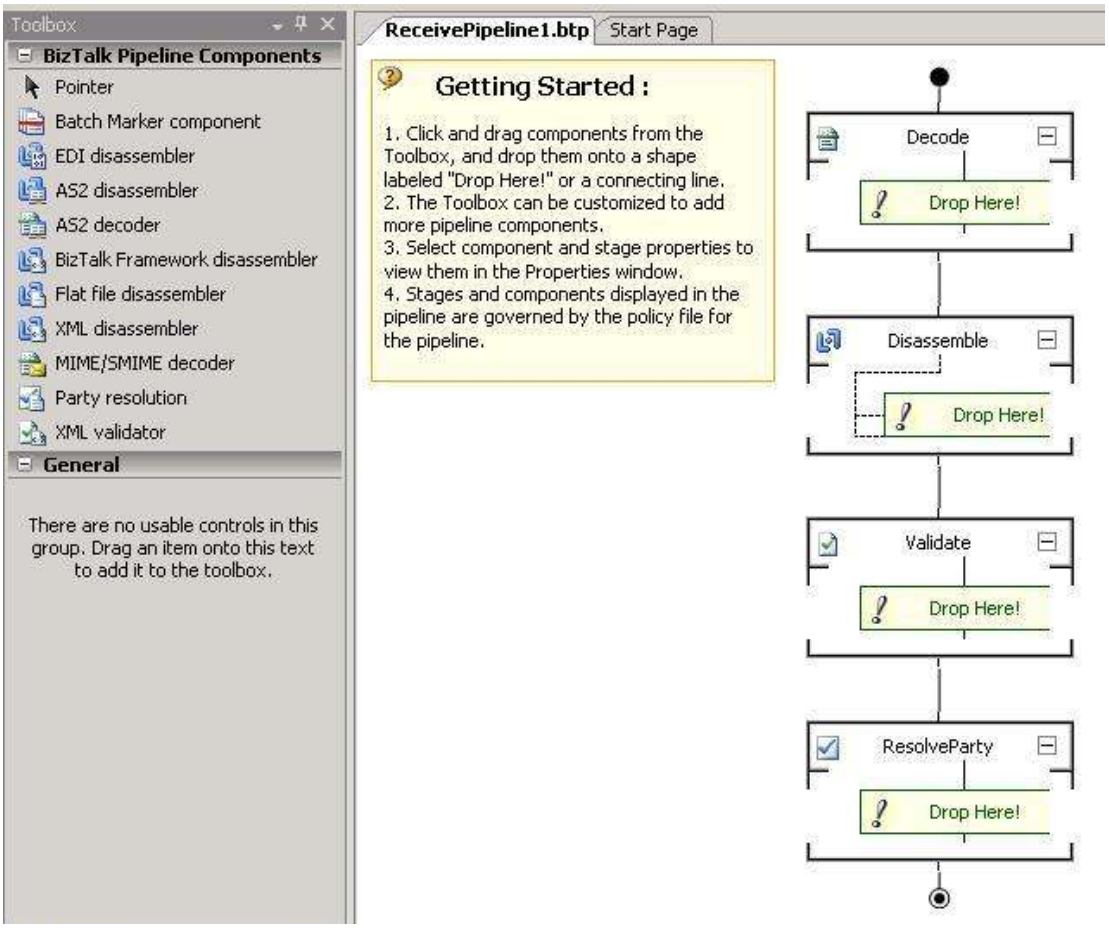
Biztalk perustuu viestien liikkeeseen eri järjestelmiin ja niiden läpi. Se myös vastaanottaa viestejä näistä järjestelmistä. Biztalkin pitää siis olla kykenevä prosessoimaan erilaisissa formaateissa olevia viestejä. Sen pitää myös pystyä toteuttamaan erilaisia protokollia, mitä vastaanottava tai lähettävä järjestelmä vaatii. Suurimman osan näistä tehtävistä Biztalkissa hoitavat pipelinet. Pipelinet ovat putkia, joiden läpi viestit kulkevat. Viestejä pystytään muokkaamaan tämän matkan varrella. Niistä voidaan myös ottaa talteen erilaisia meta-tietoja. (Jefford, Smith, Fairweather 2007, 89.)

Pipelinet liittyvät datan prosessointiin. Ne ovat vastuussa data-formaatin tunnistamisesta, datan konvertoinnista, data-formaatin oikeellisuuden varmistamisesta ja turvallisuustarkastuksista. Pipelinen toimintaan ei liity minkäänlaista varmistusta menikö viesti perille tai tuleeko vastaanottavasta järjestelmästä jotain takaisin. Tämä tehtävä kuuluu adaptereille. Jokainen Biztalkin pipeline on joko vastaanottava tai lähettävä. Kun viesti tulee Biztalkkiin, niin se viedään vastaanottavan pipelinen läpi. Kun viesti lähtee Biztalkistan, niin se viedään lähettävän pipelinen läpi. Jos Biztalkissa haluaa tehdä itse pipelineja, niin sekin on mahdollista. Biztalk Serverin 2006 versiossa tuli mukana erillinen manageri, jolla pipelineja pystyy määrittelemään. (Jefford, Smith, Fairweather 2007, 90.)

Pipelinen valinta on tärkeää siinä vaiheessa, kun määritetään projektille fyysisiä portteja. Biztalkilla on valmiina erilaisia pipelineja, joita voi määrittää portille. Monesti nämä valmiit pipelinet riittävät viestin purkamiseen käsiteltäväksi. On olemassa myös sellaisia pipelineja, jotka eivät tee viestille mitään. Ne vain päästävät viestin sellaisenaan läpi.

4.4.1 Vastaanottavan pipelinen vaiheet

Kuviossa 2 on vastaanottava pipeline. Kuviossa olevan Pipeline Managerin saa avattua Solution Explorerista valitsemalla: (add->new item->Pipeline Files->Receive Pipeline).



Kuvio 2. Vastaanottava pipeline

Vastaanottavassa pipelinessa on neljä eri vaihetta: Decode, Dissassemble, Validate ja Resolve Party. Decode-vaiheessa viesti konfiguroidaan Dissassemble-vaihetta varten. Tyypillisesti Decode-vaihetta tarvitaan, jos viestiin halutaan liittää vielä jotain. Dissassemble-vaihe koostuu kahdesta eri päätehtävästä: Viestin tunnistus ja kääntäminen Xml-muotoon sekä viestin purkaminen eri osiin ja osista edelleen moneksi eri viestiksi. Tärkeämpi näistä tehtävistä on viestin tunnistus ja kääntäminen Xml-muotoon. Viestin tunnistaminen käytännössä tarkoittaa sitä, että

Biztalk päättää mihin Schemaan viesti liittyy. Validate on vastuussa siitä, että viesti on hyvin muodostettu ja vastaa sille asetettuja Schemoja. Kaikki viestit, jotka tulevat Dissassemble-vaiheesta menevät validoinnin kautta. Biztalkin mukana tulee Xml Validator, joka suorittaa tämän tehtävän. Resolve Party vaihe selvittää, keneltä viesti tuli ja kirjoittaa viestilaatikkoon talteen tämän viestin id-tiedon. (Jefford, Smith, Fairweather 2007, 101-104.)

Pipeline-vaiheet ovat tärkeitä ymmärtää, jos Biztalkin kanssa ajautuu virhetilanteisiin. Biztalkissa viestien liikkumista pystyy seuraamaan ja pipeline-tuntemus auttaa ongelman paikantamisessa. Toisinaan tilanne voi olla se, että pitää määritellä oma pipeline jotakin tiettyä järjestelmää varten. Pipeline Managerilla on helppo luoda sellainen pipeline, joka käy ns. Flat-tiedostojen prosessointiin. Mielestäni Pipelinen tärkein tehtävä Biztalkissa on saattaa viestit siihen muotoon, että se ymmärtää ne. Tilanne on melkein sama, kuin serialisointiluokissa. Biztalkin pitää pystyä lukemaan Xml-muotoista dokumenttia, sekä joissakin tapauksissa käyttämään esimerkiksi Xpathia.

4.4.2 Lähettävän pipelinen vaiheet

Lähettävässä pipelinessa on kolme eri vaihetta: Pre-Assemble, Assemble ja Encode. Pre-Assemble-vaiheessa on mahdollista muokata viestiä, jos se ei esimerkiksi ole Xml-muotoinen. Assemble-vaihe on vastuussa kahdesta eri osa-alueesta: Viestin formaatin määrittämisestä ja sen kokoonpanon rakentamisesta. Assemble-vaihe on vastaanottavan pipelinen Dissassemble-vaiheen vastakohta. Viestin formaatti on yleensä Xml-muotoinen. Viesti saattaa tarvita erityisen Envelopen, kuten esimerkiksi SOAP-protokolla vaatii. Se voitaisiin määrittää tässä vaiheessa. Encode on viimeinen vaihe, mikä suoritetaan lähettävässä pipelinessa. Encode-vaihe enkoodaa viestit sellaisiksi, että viestit ovat mahdollisimman tietoturvallisia. (Jefford, Smith, Fairweather 2007, 106.)

4.4.3 Sisäänrakennetut pipelinet

Biztalkin asennuksen mukana tulee muutama valmiiksi määritelty pipeline. Edellä kuvatun jälkeen on varsin selvää, että monissa tilanteissa pärjää näillä valmiiksi määritetyillä työkaluilla. Ensimmäinen pipeline näistä on PassThru-pipeline. Tästä pipelinesta on kaksi eri versiota: PassThruReceive ja PassThruTransmit. Viesti kulkee vain pipelinen läpi, sille ei tehdä mitään muutoksia eikä tarkistuksia. PassThru-pipeline on hyvin tärkeä esimerkiksi SOAP-transportaatiossa. PassThru-pipelinea on hyvä käyttää aina kun se vain on mahdollista, koska se säästä tehokustannuksissa. Biztalk tarjoaa asennuksen yhteydessä myös seuraavat pipelinet: XmlReceive ja XmlTransmit. (Jefford, Smith, Fairweather 2007, 108-109.)

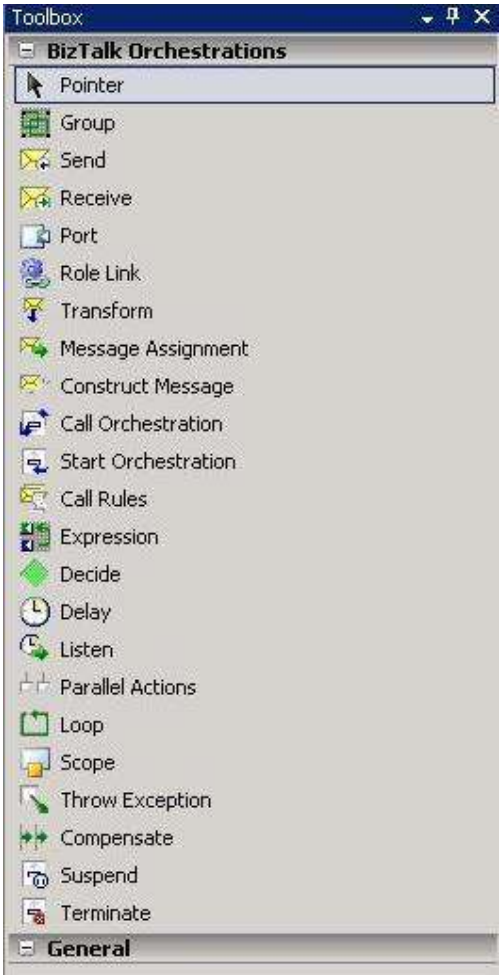
4.5 Orkestraatiot

Orkestraatiot on alun perin luotu kuvaamaan liiketoimintaprosessia. Biztalkin orkestraatio auttaa mallintamaan prosessin ilman mittavaa määrää lähdekoodia. Orkestraation perusajatus lähtee siitä, että liiketoimintaprosessin kuvaaminen on monille tahoille helpompi ymmärtää kuin ohjelmointikoodi. Liiketoimintaprosessi kuvataan erilaisilla objekteilla, kuten lähetä ja vastaanota. Lyhyesti sanottuna orkestraation tarkoitus on se, että voi keskittyä ratkaisemaan liiketoiminnan ongelmaa eikä niinkään teknistä ongelmaa. (Jefford, Smith, Fairweather 2007, 133.)

Biztalkin orkestraatio sopii moneen muuhunkin, kuin pelkästään liiketoimintaprosessien ongelmien ratkaisemiseen. Biztalk on hyvin sovellettavissa myös julkisen puolen ratkaisuihin. Orkestraatio on samalla toteutus ja kuva prosessista kaikille osapuolille. Aloittelevien IT-alan opiskelijoiden ei kannata lähteä ensimmäisenä toteuttamaan Biztalk-sovellusta. Orkestraatio saattaa näyttää kuvassa helposti toimivalta ratkaisulta, mutta sen takana on suuri määrä lähdekoodia.

Biztalk-projektin luomisen jälkeen orkestraation saa lisättyä Solution Explorerista valitsemalla: (Add->New Item->Orchestration Files->Biztalk Orchestration). Or-

kestraation luomisen jälkeen avautuu suunnittelunäkymä. Vasemmassa laidassa näkyy Toolbox, jossa on monia elementtejä, joita voi lisätä orkestraatioon. Jos Toolboxia ei näy, niin sen saa näkyviin: (View->Toolbox. Kuviossa 3 näkyy orkestraation Toolbox).



Kuvio 3. Orkestraation Toolbox

Taulukossa 1 on esitelty Toolboxin elementit ja niiden käyttötarkoitukset:

Taulukko 1. Orkestraation Elementit (Woolston 2007, 138.)

Elementti	**Selitys**

Group	Kokoaa monta elementtiä yhteen. Silloin kun orkestraatio uhkaa kasvaa liian monimutkaiseksi, niin on hyvä järjestellä elementtejä samaan ryhmään. Tämä selkeyttää prosessin mallinnusta.
Send	Käytetään viestien lähetykseen.
Receive	Käytetään viestien vastaanottoon.
Port	Mahdollistaa yhteyden viestien ja orkestraation välille.
Role Link	Abstrakti tapa valita mille osapuolelle haluaa viestin lähettää tai keneltä vastaanottaa sen.
Transform	Käytetään viestien kartoitukseen.
Message Assignment	Tämän voi laittaa Construct Message elementin sisään. Tällä elementillä voi asettaa arvoja viestille.
Construct Message	Käytetään uuden viestin luomiseen.
Call Orchestration	Käytetään, kun halutaan kutsua toista orkestraatiota synkronisesti.

Start Orchestration	Käytetään, kun halutaan kutsua toista orkestraatiota asynkronisesti.
Call Rules	Käytetään silloin kun halutaan kutsua liiketoimintasääntöjä. (Business Rules)
Expression	Tarjoaa mahdollisuuden kirjoittaa orkestraation sisään C# tyylistä koodia (Xlang), jolla voi vaikuttaa viestiin.
Decide	Antaa mahdollisuuden asettaa ehtoja orkestraatiossa viestin kululle.
Delay	Käytetään orkestraation pysäyttämiseen ennalta määritetyksi ajaksi.
Listen	Kuuntelee Delay-elementtiä ja päättää mille haaralle orkestraatiossa viestin välittää.
Parallel Actions	Käytetään jakamaan orkestraation viestin kulkua kahteen osaan.
Loop	Käytetään orkestraatiossa vastaamaan monen ohjelmointikielen While-luuppia.
Scope	Käytetään orkestraatiossa rajaamaan tietty alue. Vastaa monesti ohjelmointikielissä käytettyjä aaltosulkuja. Rajoittaa myös virheenkäsittelyn tietylle alueelle.

Throw Exception	Käytetään virheiden ”heittämiseen” niin kuin monissa ohjelmointikielissäkin.
Compensate	Antaa mahdollisuuden uudelleen asettaa mahdolliset tapahtumat.
Suspend	Käytetään orkestraation pysäyttämiseen, jos tapahtuu virhe. Viestin kulku voidaan myös käynnistää uudelleen tämän jälkeen.
Terminate	Käytetään orkestraation pysäyttämiseen, kuten Suspend-elementtiäkin, mutta viestin kulkua ei voida enää uudelleenkäynnistää.

4.5.1 Orkestraation viestit

Orkestraation viesti luodaan Orchestration View Panel -ikkunasta, jonka saa tarvittaessa näkyviin: (View->Other Windows->Orchestration View). Messages-kohdan pikavalikosta valitaan New Message. Viesti on nyt luotu ja sille pitää asettaa Schema, joka vastaa viestin rakennetta. Schema määritellään viestin ominaisuuksista. Type-kohdasta saadaan auki valikko, josta voidaan määritellä viestin Schema. Viestille voidaan myös määritellä .Net-perusteisia tietotyyppejä. (Jefford, Smith, Fairweather 2007, 166.) Näitä ei tässä opinnäytetyössä käytetty. Kaikki viestit määriteltiin Scheman avulla.

Opinnäytetyössä käytettiin viittä erilaista elementtiä: Receive, Send, Transform, Construct ja Parallel Actions. Seuraavassa esitellään nämä elementit tarkemmin.

4.5.2 Receive-elementti

Receive-elementtiä käytetään silloin kun halutaan vastaanottaa orkestraatioon viesti, joka vastaa tiettyä Schemaa. Yleisenä sääntönä voidaan pitää, että orkestraatio pitäisi aina aloittaa sellaisella Receive-elementillä, jonka ominaisuuksissa Activate-arvoon on määritelty True. Monesti Activate-arvon muuttaminen unohtuu ja silloin orkestraatio antaa seuraavanlaisen virheen:

“you must specify at least one already-initialized correlation set for a non-activation receive that is on a non-selfcorrelating port”

Receive-elementti siirretään ensiksi orkestraatioon ja sen jälkeen sille pitää määritellä viesti. Receive-elementiltä löytyy ominaisuuksista kohta Message. Sinne pitää määritellä viesti, mitä elementti käyttää. (Jefford, Smith, Fairweather 2007, 169-170.)

4.5.3 Send-elementti

Send-elementti on tarkoitettu viestien lähettämiseen. Viestin lähettämisen jälkeen orkestraatio jää odottamaan vastausta. Tämä vastaus siirretään Receive-elementille. Send-elementille pitää myös määritellä viesti, jonka se lähettää. (Jefford, Smith, Fairweather 2007, 170.)

Biztalkissa orkestraatiot toimivat ”lähetä ja vastaanota” -periaatteella kuten Internet. Tämä ajattelutapa on hyvä silloin kun suunnittelee orkestraation toteuttamista. Viestin voi toki lähettää kohdejärjestelmään ilman minkäänlaista vastaanottotapaa. Silloin on kuitenkin mahdollista, että viestin perille meno vaarantuu. Hyvä tapa on aina varmistaa edes http-koodin 200 avulla, että viestin lähetys onnistui ja se vastaanotettiin onnistuneesti.

4.5.4 Parallel-elementti

Parallel-elementti on tarkoitettu liiketoimintaprosessien jakamiseen. Kehittäjän näkökulmasta saattaa näyttää, että useita tehtäviä toteutetaan samanaikaisesti. Oikeastaan Parallel-elementin viestivirta menee siinä järjestyksessä, kun ne suoritetaan. Toinen haara jää odottamaan toista, jos toteuttamisessa menee kauemman aikaa. Tämä elementti on hyvä esimerkiksi silloin, kun orkestraatioon tulee parametri, joka pitää saada kahteen tai useampaan järjestelmään. Parametri voidaan tässä tapauksessa jakaa kahdelle tai useammalle eri viestivirralle. Parallel-elementin avulla on myös hyvä jakaa orkestraatiota selkeämmäksi. (Woolston 2007, 138.)

4.5.5 Construct- ja Transform-elementit

Biztalkin viestejä ei voi muuttaa silloin, kun ne ovat viestilaatikossa. Tämän vuoksi on olemassa Construct-elementti, jolla voi kloonata viestin datan ja näin muuttaa sisältöä. Transform-elementti on Construct-elementin lapsi. Tämä tarkoittaa sitä, että Transform-elementin voi siirtää Construct-elementin sisälle. Tällainen määrittely vaatii Biztalk Mapper -managerin. Construct-elementtiin pitää aina määritellä tuleva viesti ja sieltä lähtevä viesti. (Woolston 2007, 138.)

Transform-elementti on Construct-elementin lapsi. Transformaation jälkeen viesti muuttaa muotoaan kartoituksen johdosta. Kartoitus hoidetaan Biztalk Mapper -managerin avulla. Tällä elementillä on helppo siirtää dataa toisesta viestistä toiseen. Elementti vaatii aina määrittelyn, mitkä viestit toimivat lähdedatana ja mitkä vastaanottavana datana. (Woolston 2007, 138.)

Construct-elementti esitellään ensin ja sille määritellään tarvittavat ominaisuudet. Tämän jälkeen Construct-elementin sisälle viedään Transform-elementti. Lopuksi käynnistetään Biztalk-Mapper, jolla tehdään vaadittavat kartoitukset.

4.5.6 Transformaatiot ja Biztalk-kartoitukset

Viestin transformaatio tarkoittaa prosessia, jossa alkuperäinen viesti konvertoidaan uuden tyyppiseksi viestiksi. Yleensä viestin koko kasvaa tämän prosessin aikana. Transformaatio on erittäin keskeinen, mutta yksinkertainen osa Biztalk Serveriä. Se on tärkeä osa järjestelmäintegraatiota, koska tällä tapaa on mahdollista ottaa toisesta järjestelmästä viesti ja muuttaa se toiseen järjestelmään sopivaan muotoon. (Jefford, Smith, Fairweather 2007, 211.)

Biztalk-kartoittaja (Mapper) tarjoaa graafisen esitystavan transformaatioille. Kartoittaja siirtää datan yhdestä tai useammasta sisään tulevasta Xml-viestistä lähtevään Xml-viestiin. Kartoittaja perustuu Xslt-transformaatioon, joka käyttää .Net XslTransform-luokkaan. Biztalk Server 2006 -versiossa on parempi tuki suurien viestien transformaatiolle kuin aikaisemmissa versioissa. Kartoittaja tarjoaa myös monenlaisia tapoja muuntaa dataa ja asettaa sille ehtoja. Näitä funktioita voi lisätä kartoittajalle. Ne voidaan siirtää samalla tavalle kartoitettavalle alueelle, kuten orkestraation elementitkin. (Jefford, Smith, Fairweather 2007, 212.)

4.6 Loogiset portit

Orkestraatio tarvitsee loogisia portteja viestien reitittämiseen. Porttien avulla määritellään, mistä viestit tulevat sisään orkestraatioon ja mistä ne lähtevät. Receive-elementin luonnin ja konfiguroinnin jälkeen se tarvitsee loogisen portin, jonka avulla viesti pääsee sisään orkestraatioon. Loogiset portit ovat vain esitystapa viestin tulo- ja lähtöpaikoille. Niillä ei ole mitään tekemistä fyysisen sisään- tai ulosmeno reitin kanssa. Loogiset portit määrittelevät vain kommunikaatiokaavan ja viestin tyypin. Silloin kun orkestraatiossa määritellään loogiset portit, niin adaptereista ei tarvitse välittää vielä mitään. Tämä antaa joustavuutta sovelluksen kehittämiseen, koska logiikka voidaan määritellä ennen fyysistä toteutusta. Loogiselle portille pitää määritellä minkälaista kommunikaatiokaavaa se suorittaa ja viestin tyyppi. Biztalkilla on seuraavat kommunikaatiokaavat valittavana: One Way, Solicit/Response ja Request/Response. Kaavan valintaan vaikuttaa paljon se mitä adapteria tullaan käyttämään. File-adapteri hyväksyy vain One Way -kaavan kun taas SOAP-adapteri tukee kaikkia kommunikaatiokaavoja. Kommunikaatiokaavan valinta riippuu sovelluksen tarpeista. Silloin kun haluaa lähettää vastauksen lähdejärjestelmään, niin kannattaa käyttää Request/Response-kaavaa. Silloin kun haluaa vain vastaanottaa viestin, niin silloin kannattaa käyttää One Way -kaavaa. (Jefford, Smith, Fairweather 2007, 153)

Kuviossa 4 on orkestraationäkymä. Port Surface -alustoille voidaan luoda loogisia portteja. Sillä kummalle puolelle portin sijoittaa, ei ole ohjelmallista väliä. Pylväät ovat olemassa molemmilla puolilla, siksi että orkestraatioon saa luotua selkeyttä, kun viivat eivät mene toistensa päältä. Portti luodaan kuviossa 4 esitetyllä tavalla. Luonti onnistuu myös esimerkiksi raahaamalla Port-elementti alueelle. New Configured Port -tapa on helpompi, koska se sisältää askel askeleelta opastuksen.

Kuvio 4. Loogisen portin luonti orkestraationäkymässä

Portille pitää valita nimi. Nimi voi olla mikä tahansa, mutta hyvänä tapana voidaan pitää sitä, että ei käytetä skandeja. Seuraavaksi pitää valita portin tyyppi, haluttu kommunikaatiokaava sekä rajoitukset. Porttityypin asetukset jaetaan projektissa, joten portin tyypin voi määrittää jo olemassa olevan tyypin mukaan. Porttityypin rajoituksia on olemassa kolme: Private, Internal ja Public. Nämä määritykset periytyvät .Net-ohjelmointikielien määrityksistä. Private määrittelee portin, niin että sitä voi käyttää vain kyseisessä orkestraatiossa. Internal-määrityksellä portti saadaan näkymään muissakin projektin orkestraatioissa. Public määrittelee portin, niin että sitä voi käyttää kuka tahansa. Internal toimii monessa tapauksessa. Public määritys on välttämätön, jos portti on tarkoitettu toimimaan Webservicen kanssa. (Jefford, Smith, Fairweather 2007, 154-155.)

Viimeisessä portin konfigurointi-ikkunassa suoritetaan sidonta. Sidonta määritellään yleensä ”Specify Later” -määreellä. Määre on suositeltava siksi että loogiset portit kannattaa sitoa fyysisiin vastaaviin vasta silloin, kun orkestraatiota ollaan käynnistämässä. Tässä vaiheessa pitää myös määrittää kommunikaatiokaavan suunta. Kommunikaatiosuunnalle on aina kaksi vaihtoehtoa.

4.7 Fyysiset portit

Fyysisillä porteilla Biztalk pystyy lähettämään ja vastaanottamaan dataa. Porteilla määritellään, miten Biztalkin tulee kommunikoida eri rajapintojen kanssa. Kun käytetään orkestraatioita, niin loogiset ja fyysiset portit sidotaan toisiinsa. (Jefford, Smith, Fairweather 2007, 39.)

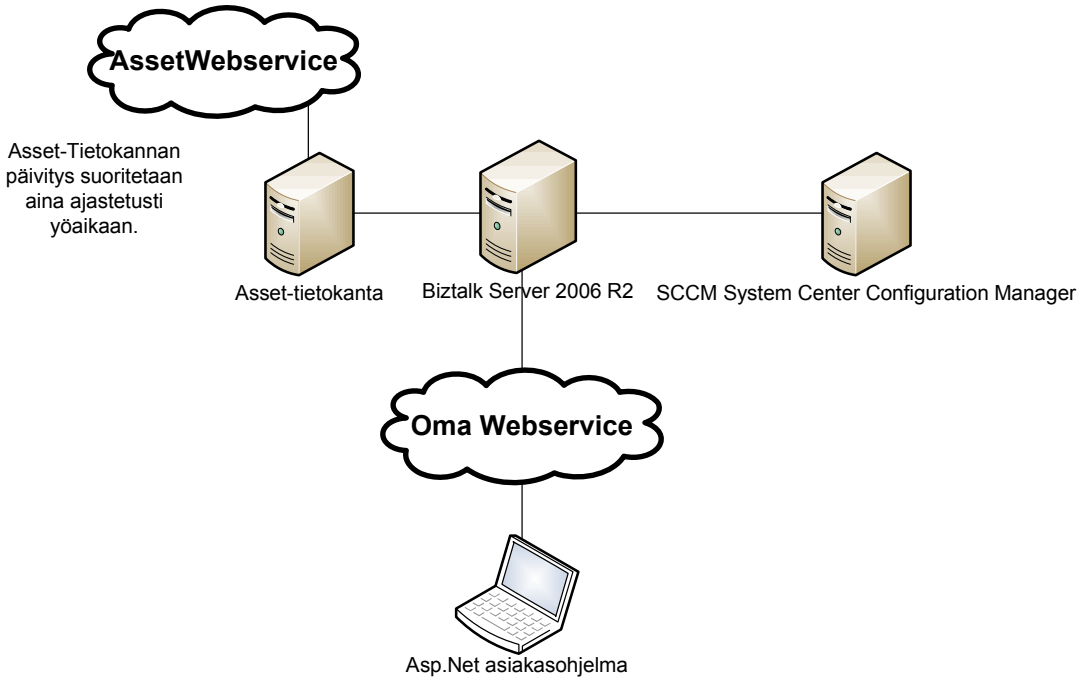
Vastaanottavat portit ja vastaanottopaikat: Biztalkissa vastaanottava portti määritellään ensin ja sen jälkeen vastaanottopaikka. Portin määrittäminen onnistuu Biztalk Explorerin kautta. Portin määrityksistä pakollinen tieto on vain portin nimi. Porttiin voi määritellä erikseen erilaisia kartoituksia. Kartoituksia voi siis hoitaa jo silloin kun viesti tulee sisään Biztalkiin. Porttiin voi myös määritellä autentikoinnin. Vastaanottopaikan määritteleminen on tärkeä osa viestin vastaanottoa. Tälle paikalle määritellään pipelinet, adapterit ja konkreettinen vastaanottopaikka. Vastaanottopaikalle pitää myös määritellä kommunikaatiokaava eli onko se One-Way vai Request-Response. Vastaanottopaikka voi olla esimerkiksi URL tai tiedostojärjestelmän polku. Pipeline kohtaan voi valita: XmlReceive tai PassThru. Adapterien kohdalla valinnan voi tehdä seuraavista adaptereista: File, Ftp, http, Mqseries, MSQM, POP3, SOAP, SQL tai sitten jonkin WCF-adaptereista.

Lähettävät portit: Lähettävissä porteissa ei tarvitse määritellä kuin itse portti. Lähettävä portti lähettää viestin toiseen järjestelmään tai tietolähteeseen. Portin määrittäminen onnistuu Biztalk Explorerin kautta. Portille pitää määritellä nimi, adapteri ja pipeline. Sille pitää myös aluksi määrittää kommunikaatiokaava eli onko se Static One-Way vai Static Solicit-Response.

5 LAITEREKISTERIEN INTEGROINTI

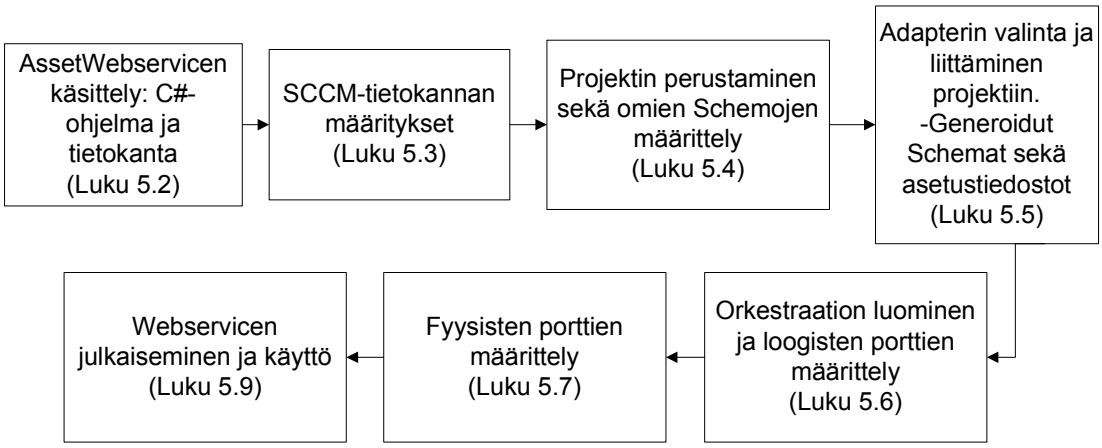
Toteutuksen tavoitteena oli integroida kaksi eri laiterekisteriä Biztalkin avulla. Ensimmäinen laiterekisteri on 3stepIT:n laiterekisteri, jonka asiakas Seinäjoen ammattikorkeakoulu on. Toinen laiterekisteri on Seinäjoen ammattikorkeakoulun oma laiterekisteri SCCM-tietokanta. Yhteiseksi tekijäksi näiden tietolähteiden välille valittiin laitteen sarjanumero.

Kuviossa 5 on kuvattu projektin arkkitehtuuri. AssetWebserviceen ei ollut mahdollista ottaa suoraan yhteyttä Biztalkin avulla. Tämän vuoksi sille luotiin oma tietokanta, joka päivitetään öisin kokonaan.



Kuvio 5. Esimerkkiprojektin arkkitehtuuri

5.1 Toteutusesimerkin prosessi



Kuvio 6. Esimerkkiprojektin prosessi.

Kuvion 6 prosessikaavioon on nimetty keskeiset eri vaiheet sekä mainittu vaihetta vastaava opinnäytetyön luku.

5.2 Asset

Asset-puolen integraatio oli haasteellinen, koska se ei toteuttanut minkäänlaista Webservicen kuvauskieltä. Asset-puolen dokumentaatiossa oli alkeellinen Java-ohjelma, joka suorittaa pyynnön. Tässä opinnäytetyössä kehitettiin automatisoidumpi ohjelma C#-kielellä, joka tekee myös päivityksen tietokantaan. Alussa suuria hankaluuksia tuotti myös se, että Seinäjoen ammattikorkeakoulun tunnuksien aktivointi puuttui.

5.2.1 AssetWebService

Opinnäytetyössä ei voitu käyttää suoraan tätä Webservice-rajapintaa Biztalkin kanssa, koska AssetWebService ei toteuta WSDL-kuvauskieltä, joka on tarkoitettu virallisten Webservice-palveluiden kuvaamiseen. AssetWebservice toimii niin, että sille lähetetään String-muotoisena merkkijonona Xml-tiedosto ja salasana Md5-

salattuna http-post-metodin avulla. Tässä opinnäytetyössä tärkeintä oli hakea palvelulta kaikista laitteista tiedot. Tietojen hakemiseen käytettiin seuraavanlaista Xml-dokumenttia.

Esimerkki 4. AssetWebService Request Xml-dokumentti

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<export-request response="xml" id="1100018">
<customer>
<id>1100018</id>
</customer>
<exportset>
 <export rowlevel="device" exportid="2">
<columnset>
<column name="name" code="name"/>
<column name="deviceid" code="deviceid"/>
<column name="serialnumber" code="serialnumber"/>
<column name="integrationid" code="integrationid"/>
<column name="username" code="username"/>
<column name="productgroupname" code="productgroupname"/>
<column name="processor" code="processor"/>
<column name="memory" code="memory"/>
<column name="displaysize" code="displaysize"/>
<column name="storage" code="storage"/>
<column name="extratext1" code="extratext1"/>
<column name="extratext2" code="extratext2"/>
<column name="extranumber1" code="extranumber1"/>
<column name="extranumber2" code="extranumber2"/>
<column name="costcenter" code="costcenter"/>
<column name="location" code="location"/>
<column name="version" code="version"/>
<column name="extrainfo" code="extrainfo"/>
<column name="project" code="project"/>
<column name="companyproductgroup" code="companyproductgroup"/>
<column name="extrainfo1" code="extrainfo1"/>
<column name="extrainfo2" code="extrainfo2"/>
<column name="deviceprice" code="deviceprice"/>
<column name="devicerent" code="devicerent"/>
<column name="contractnumber" code="contractnumber"/>
<column name="length" code="length"/>
<column name="startdate" code="startdate"/>
<column name="enddate" code="enddate"/>
<column name="deliverydate" code="deliverydate"/>
<column name="signingdate" code="signingdate"/>
```

```
<column name="interval" code="interval"/>
<column name="contractprice" code="contractprice"/>
<column name="terminated" code="terminated"/>
<column name="endingoption" code="endingoption"/>
<column name="publicstatus" code="publicstatus"/>
<column name="changedate" code="changedate"/>
</columnset>
</export>
</exportset>
</export-request>
```

AssetWebServicen Request-dokumentissa käytetään vanhempaa ISO-8859-1-koodaustapaa. Uudempi Utf-8 olisi suositeltava. Opinnäytetyössä piti käyttää kyseistä koodaustapaa mahdollisten virhetilanteiden välttämiseksi. Prologin jälkeen tulee ensimmäinen elementti Export-Request, jonka attribuutteihin pitää määritellä tiedostomuoto sekä asiakas-id. Tiedostomuoto-kohtaan voi määritellä myös ”Zip”, jolloin AssetWebService palauttaa vastauksen zip-pakattuna. Asiakas-id saadaan 3stepIT:ltä. Asiakas-id pitää määritellä myös Customer-elementtien sisään. Export-elementin attribuutteihin pitää määrittää tekeekö haun laitetasolta vai sopimustasolta. Device tarkoittaa laitetasoa. Laitetason id on tällöin 2. Columnset-elementin sisällä esitellään ne tiedot, joita halutaan hakea AssetWebserviceltä. Name-attribuuttiin voi määritellä haluamansa palautusnimen. Code-attribuutti on AssetWebServicen määräämä nimi tietylle tiedolle. Seuraavassa esimerkki siitä minkälainen AssetWebServicen palauttama Xml voi olla.

Esimerkki 5. AssetWebService Response Xml-dokumentti

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE import-request SYSTEM "https://asset-3stepit.com/asset/Asset-
files/export_response.dtd">
<export-response id="1100018" >
 <exporttime>2009-08-14 17:04</exporttime>
<export-status code="1" >
<message>All ok.</message>
</export-status>
<exportset>
<export rowlevel="device" status="1" exportid="1" >
<message>All ok</message>
```

```
<entity>
<attribute code="name" >PY TX150S6R</attribute>
<attribute code="deviceid" >160549967</attribute>
<attribute code="serialnumber" >YK8B036669</attribute>
<attribute code="integrationid" />
<attribute code="username" >KURIKKA / Laitinen Matti</attribute>
<attribute code="productgroupname" >Servers</attribute>
<attribute code="processor" />
<attribute code="memory" />
<attribute code="displaysize" />
<attribute code="storage" />
<attribute code="extratext1" assetname="ExtraText 1" />
<attribute code="extratext2" assetname="ExtraText 2" />
<attribute code="extranumber1" assetname="Korjataan" >0</attribute>
<attribute code="extranumber2" assetname="Korvataan" >0</attribute>
<attribute code="costcenter" >90 Koulutuskeskus Sedu, Kurikka</attribute>
<attribute code="location" />
<attribute code="version" />
<attribute code="extrainfo" assetname="Extra Info" />
<attribute code="project" />
<attribute code="companyproductgroup" />
<attribute code="extrainfo1" assetname="Extra Info 1" />
<attribute code="extrainfo2" assetname="Extra Info 2" />
<attribute code="deviceprice" >2290</attribute>
<attribute code="devicerent" >151.74</attribute>
<attribute code="contractnumber" >007258-AD01-092-000</attribute>
<attribute code="length" >48</attribute>
<attribute code="startdate" >2009-07-01</attribute>
<attribute code="enddate" >2013-06-30</attribute>
<attribute code="deliverydate" />
<attribute code="signingdate" >2009-05-20</attribute>
<attribute code="interval" >3</attribute>
<attribute code="contractprice" >9160</attribute>
<attribute code="terminated" >0</attribute>
<attribute code="endingoption" >6</attribute>
<attribute code="publicstatus" >20</attribute>
<attribute code="changedate" >2009-05-20</attribute>
</entity>
</export>
</exportset>
</export-response>
```

5.2.2 C#-ohjelma tietokannan päivitykseen

Esimerkissä 6 esitellään ohjelma, jolla haetaan kaikki tiedot laitteista AssetWeb-Serviceltä. Ohjelma on konsoliohjelma, jossa ei ole käyttöliittymää. Ohjelma ei tarvitse sitä, koska se on ajastettu suoritettavaksi aina yöaikaan. Asset-tietokanta päivitetään yöaikaan tietoliikenneruuhkien välttämiseksi. Ohjelma on jaettu pääohjelmaan ja neljään eri metodiin.

Esimerkki 6. C#-pääohjelma ja kirjastot

```
using System;
//Ladataan kirjastoja kokoelmia varten
using System.Collections.Generic;
using System.Collections.Specialized;
//Ladataan kirjasto relaatiotietokantoja varten
using System.Data.Linq;
//Ladataan kirjasto tekstin lukua varten
using System.Text;
//Ladataan kirjastoja salauksia ja sertifikaatteja varten
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
//Ladataan pari kirjastoja Xml-tiedoston käsittelyyn
using System.Xml;
using System.Xml.XPath;
//Ladataan kirjasto tiedostonkäsittelyä varten
using System.IO;
//Ladataan kirjastoja verkkokäytettävyyttä varten
using System.Web;
using System.Net;
using System.Net.Security;
//Ladataan kirjasto, jolla voidaan tehdä asetuksia
using System.Configuration;
//Ladataan kirjastoja tietokantaoperaatioita varten
using System.Data.Sql;
using System.Data.SqlClient;

namespace SqlDumpAsset
{
    class Program
    {
        static void Main(string[] args)
        {
            //Asetetaan 3stepIt:n tarjoama salasana muuttujaan
            string password = "*********";
            //Asetetaan tiedoston sijaintipolku muuttujaan
            string filepath = "c:\\assetwebservice\\export_request.xml";
            //Luetaan Xml-tiedosto muuttujaan tiedoston sijaintipolun
           avulla
            string xmlstring = readXml(filepath);
           //Generoidaan MD5-salattu avain Xml-tiedostosta, sekä
           salasanasta
```

```
            string key = MD5Encrypt(xmlstring + password);
             //Asetetaan 3stepIt:n tarjoama URL ja generoitu avain
            muuttujaan
            string url =
            "https://87.94.3.43/integration/assetwebservice?key=" + key;
            /*Suoritetaan HTTP-POST ja vastaanotetaan vastaus URL-
            osoitteen ja
            Xml-tiedoston avulla*/
            string xmlresponse = PostAndGetResponse(url, xmlstring);
            /*Parsitaan vastauksesta oleelliset tiedot ja tehdään
            tietokantapäivitys*/
            ParseXmlAndDoDBupdate(xmlresponse);
        }
```

Ohjelmassa on kirjastokutsut, joita tarvitaan koko ohjelman suorituksen ajan. Kirjaston lisääminen ei välttämättä aina auta, kun kehitetään C#-ohjelmaa. Joissakin tapauksissa kirjasto pitää lisätä ”manuaalisesti” referenssiksi projektille Solution Explorerista Add Reference -komennolla. Luettelosta voi valita haluamansa referenssin. Pääohjelma jakautuu neljään eri metodiin. Neljä eri metodia ovat: Xml-tiedoston luku String-muuttujaan, MD5-salausavaimen generointi, http-Post-metodin suoritus ja Tietokantapäivityksen suorittaminen. Salasana, Url ja Xml-tiedosto saatiin 3stepIT:ltä. Xml-tiedostoa piti muokata omiin tarpeisiin, jotta kaikki laitteet saadaan vastauksena AssetWebServiceltä.

Metodissa ”ReadXml” Xml-tiedosto luetaan kuten mikä tahansa tekstitiedosto. Tässä metodissa käytettiin Stringbuilder-objektia Xml-tiedoston oikean muodon varmistamiseksi. AssetWebService tarvitsee esimerkiksi tiedostossa olevat rivinvälit. Metodi tarvitsee parametrikseen tiedoston sijainnin. Metodi lukee tiedoston rivi kerrallaan ja palauttaa tiedoston String-merkkijonona kutsujalle. Täydellinen kuvaus metodin toiminnasta on liitteessä 1.

Metodissa ”MD5Encrypt” generoidaan MD5-salattu avain, joka lähetetään AssetWebServicelle URL-osoitteen mukana. Ilman avainta AssetWebService ei pysty tunnistamaan asiakasta. Tärkeimmät vaiheet avaimen generoinnissa ovat:

1. MD5-hash-avaimen konvertointi Byte-muuttujaan

2. Base64-konvertointi avaimelle

3. Avaimen URL-enkoodaus.

Metodi syötteenä on String-merkkijono, jossa on yhdistettynä Xml-tiedosto String-merkkijonona ja 3stepIT:ltä saatu salasana. Metodi palauttaa näistä muodostetun avaimen. Täydellinen kuvaus metodin toiminnasta on liitteessä 2.

Metodissa ”PostAndGetResponse” suoritetaan http-POST-pyyntö ja vastaanotetaan sen jälkeen vastaus. Tärkeimmät vaiheet metodissa ovat: pyynnön lähetys, pyynnön vastaanotto ja SSL-sertifikaatin ohitus esimerkin 7 mukaisesti. Sertifikaatin ohitus piti tehdä ohjelmaan, koska AssetWebServicen sertifikaatti ei ole sääntöjen mukainen. Microsoft edellyttää, että sertifikaatit ovat aitoja. Sertifikaattien tarkistus poistetaan käytöstä ServicePointManagerin kautta. Metodi jakautuu kahteen vaiheeseen:

1. Metodi ottaa vastaan URL-osoitteen, johon on liitettynä MD5-salattu avain sekä Xml-merkkijonon.

2. Metodi palauttaa AssetWebServiceltä saadun string-virran ja poistaa sitä tyhjät kohdat.

Täydellinen kuvaus metodin toiminnasta on liitteessä 3.

Esimerkki 7. Sertifikaatin ohitusmetodi

```
//Asetetaan kaikki sertifikaatit hyväksytyiksi
public static bool TrustAllCertificatesCallback(object sender,
X509Certificate cert, X509Chain chain, SslPolicyErrors errors)
{
    //Palautetaan true
    return true;
}
```

Metodissa ”ParseXmlAndDoDBupdate” käydään läpi vastaanotettu Xml ja lisätään siitä tiedot tietokantaan. Xpathia käytetään tässä metodissa ehtojen luomisessa

niille tiedoille, jotka halutaan parsia Xml-dokumentilta. Ennen tietojen lisäämistä taulut tyhjätään tiedoista. Metodi vastaanottaa parametrina Xml-vastauksen. Metodi ei palauta mitään. Se kirjoittaa vain väliaikatietoja ajoaikana. Näitä tietoja tarvitaan muuten vain testausvaiheen aikana. Muina aikoina ohjelma ajetaan itsenäisenä, jolloin testiviestejä ei tarvita. Täydellinen kuvaus metodin toiminnasta on liitteessä 4.

5.2.3 C#-ohjelman suorituksen ajastaminen

C#-ohjelman ajastetaan ohjauspaneelista: Scheduled Tasks->Add Scheduled Task.

1. Valitaan mikä ohjelma halutaan ajastaa. Valitaan (Browse) ja haetaan C#-ohjelman projektikansio. Ajettava exe-tiedosto löytyy hakemistosta: (bin->Debug).

2. Määritellään milloin ohjelma suoritetaan: päivittäin, viikottain, kuukausittain, yhden kerran, tietokoneen käynnistyessä vai silloin kun kirjaudutaan sisään. Tässä tapauksessa ohjelma ajastettiin päivittäin ajettavaksi.

3. Valitaan alkamisaika ja päivämäärä. Valikosta voidaan myös ajastaa ohjelma suoritettavaksi, vaikka pelkästään arkipäivinä.

4. Asetetaan käyttäjätunnus ja salasana. Tämä tehdään siksi, että saadaan varmistettua käyttäjän oikeudet ohjelman suoritukseen.

Seuraavaksi pitää tehdä vielä yksi muutos avataan: (Advanced Settings->Settings). Valitaan ”Wake the computer to run this task” aktiiviseksi. Windows ei suorita ollenkaan ohjelmaa, jos tämä asetus ei ole voimassa. Tämä johtuu siitä, että konsoliohjelma ei itsessään sisällä minkäänlaista herätystoimintoa. Nyt ohjelman pitäisi näkyä yhtenä objekteista ”Scheduled Task” -ikkunassa.

5.2.4 Tietokantarakenne ja proseduuri

Kuviossa 6 on Asset-tietokannan taulut

AssetData	
PK	**asset_id**
	name
	idnumber
	serialnumber
	integrationid
	username
	productgroupname
	processor
	memory
	displaysize
	storage
	extratext1
	extratext2
	extranumber1
	extranumber2
	costcenter
	location
	version
	extrainfo
	project
	companyproductgroup
	extrainfo1
	extrainfo2
	deviceprice
	devicerent
	contractnumber
	length
	startdate
	enddate
	deliverydate
	signingdate
	interval
	contractprice
	terminated
	endingoption
	publicstatus
	changedate

Updated	
PK	**DateId**
	LastUpdated

Kuvio 7. Asset-tietokannan rakenne

Rakenne määriteltiin yksikertaiseksi, niin että toinen AssetData-taulu toimii kaiken laitteeseen liittyvän tiedon varastona ja Updated-tauluun kirjataan päivitysaika. Tietokannan nimenä toimii sama ”AssetData”, kuin taulunkin nimessä. Näille kah-

delle taululle piti määrittää proseduuri, joka hakee kaiken tiedon tauluista laitteen sarjanumeron avulla. Proseduuri pitää määritellä silloin, kun halutaan kommunikoida Biztalkista SqlServerin kanssa. Seuraavassa esitellään kyseinen proseduuri.

Esimerkki 8. Asset-tietokantaproseduuri

```
USE [AssetData]
GO
/****** Object:  StoredProcedure [dbo].[sp_GetAssetDevice]    Script
Date: 08/20/2009 12:01:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[sp_GetAssetDevice]
@serialnumber varchar(50)
As
Select a.*,u.LastUpdated
From AssetData as a,Updated as u

Where a.serialnumber = @serialnumber
```

Proseduurin syöteparametrina on sarjanumero, jonka pituus on maksimissaan 50 merkkiä. Proseduuri hakee kaiken tiedon AssetData-taulusta ja päivitysajan Updated-taulusta. Yllä oleva esimerkki on skripti, jonka SqlServer generoi automaattisesti. Proseduurin voi suorittaa esimerkiksi komennolla: ”`EXEC sp_GetAssetDevice @serialnumber='145491-SEI-0704AU029'`”.

5.3 SCCM-tietokanta

SCCM (System Configuration Manager) on Microsoftin julkaisema järjestelmänhallintatuote. SCCM-ohjelmistolla pystytään hallitsemaan isoja laitekokonaisuuksia verkon yli. Esimerkiksi päivitykset voidaan asentaa kaikille koneille yhdellä kertaa. (Ilenius 2009, 3.)

Seinäjoen ammattikorkeakoulu käyttää tätä ohjelmistoa laitteiden hallintaan. Tässä opinnäytetyössä haetaan tietoja laitteista SCCM-tietokannasta sarjanumeron avulla. SCCM-tietokanta on Microsoft Sql Serverissä, joten dataa oli luonnollista hakea

Biztalkin avulla. SCCM-tietokannan kohdalla keskityttiin hakemaan BIOS-tietoja laitteista. Tietokantaan ei voi tehdä muutoksia. Esimerkissä 9 esitellään proseduuri, jolla haetaan tietoja SCCM-tietokannasta Biztalkiin.

Esimerkki 9. SCCM-proseduuri

```
USE [SMS_DPS]
GO
/****** Object:  StoredProcedure [dbo].[sp_GetSMSDevice]    Script Date:
08/21/2009 17:48:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_GetSMSDevice]
(@Seriaali varchar(100))
AS
SELECT * FROM v_GS_PC_BIOS


WHERE SerialNumber0 Like @Seriaali
```

Proseduuri ottaa parametrina vastaan sarjanumeron, jonka pituus on maksimissaan 100 merkkiä. Yllä oleva esimerkki on skripti, jonka SqlServer generoi automaattisesti. Proseduurin voi suorittaa esimerkiksi komennolla: `EXEC sp_GetSMSDevice @Seriaali='YK3N066626'.`

5.4 Projektin perustaminen

Biztalk projekti perustetaan seuraavalla tavalla: (File->New->Project->Biztalk Projects->Empty Biztalk Server Project). Projektin luonnin yhteydessä, sille asetetaan oletuksena olevat Biztalk-kirjastot: Microsoft.Biztalk.DefaultPipelines, Microsoft.Biztalk.GlobalPropertySchemas, System, System.configuration ja System.XML. Projektin nimeksi kannattaa valita jokin sitä kuvaava sana tai niiden yhdistelmä. Tässä projektissa nimeksi valittiin: ”Integration_Asset_SMS”.

5.4.1 Omat Schemat

Tämä Biztalk-projekti tarvitsee kaksi itse määriteltyä Schemaa. Ensimmäinen välittää parametrin orkestraatioon ja toisen mukana viedään vastaus Webservicen kutsujalle. Scheman lisääminen projektiin onnistuu seuraavanlaisesti: (Add->New Item->Schema Files->Schema). Ensimmäisen Scheman nimeksi asetettiin: ”Serialnumber”.

Esimerkki 10. Serialnumber-Schema

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
xmlns="http://Integration_Asset_SMS.Serialnumber" targetNames-
pace="http://Integration_Asset_SMS.Serialnumber"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Parameter">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Serialnumber" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Scheman enkoodaus on ”utf-16” ja Xml-versio on ”1.0”, jotka ovat Biztalkissa oletuksena. Schemalla on neljä eri nimiavaruutta. ”http://schemas.microsoft.com/BizTalk/2003” on Biztalkin oma nimiavaruus, joka generoidaan Schemaan automaattisesti. Nimiavaruus määrittää Scheman Biztalkissa kulkevaksi viestiksi. Toinen automaattisesti generoitu nimiavaruus on: ”http://www.w3.org/2001/XMLSchema”. Tämä nimiavaruus kertoo sen, että Schema kuuluu W3C-Consortiumin standardiin. Scheman nimiavaruus eli ”targetNamespace” pitää määritellä itse. Tämä nimiavaruus ei saa olla sama, kuin muilla viesteillä Biztalk-projektissa. Tähän Schemaan määriteltiin elementti: ”Parameter”, jonka lapsielementtinä toimii ”Serialnumber”. ”Serialnumber” on xs:string-tyyppinen elementti. Kyseinen Schema on tehty Biztalkin Schema-editorilla. Editori tukee Scheman suunnittelua ja estää kirjoitusvirheet tärkeissä määrittelyissä.

Toisen itse määritellyn Schema nimeksi asetettiin ”FinalAnswer”.

Esimerkki 11. FinalAnswer-Schema

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
xmlns="http://Integration_Asset_SMS.FinalAnswer" targetNames-
pace="http://Integration_Asset_SMS.FinalAnswer"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Final">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="SMS_Answer">
     <xs:complexType>
      <xs:sequence>
       <xs:element minOccurs="0" name="ResourceID" type="xs:string" />
       <xs:element minOccurs="0" name="Timestamp" type="xs:string" />
       <xs:element minOccurs="0" name="Description" type="xs:string" />
       <xs:element minOccurs="0" name="Manufacturer" type="xs:string" />
       <xs:element minOccurs="0" name="ReleaseDate" type="xs:string" />
       <xs:element minOccurs="0" name="SerialNumber" type="xs:string" />
       <xs:element minOccurs="0" name="SmsBiosVersion" type="xs:string" />
       <xs:element minOccurs="0" name="Software" type="xs:string" />
       <xs:element minOccurs="0" name="Version" type="xs:string" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Asset_Answer">
     <xs:complexType>
      <xs:sequence>
       <xs:element minOccurs="0" name="Name" type="xs:string" />
       <xs:element minOccurs="0" name="Idnumber" type="xs:string" />
       <xs:element minOccurs="0" name="Serialnumber" type="xs:string" />
       <xs:element minOccurs="0" name="Username" type="xs:string" />
       <xs:element minOccurs="0" name="ProductGroupName" type="xs:string" />
       <xs:element minOccurs="0" name="Processor" type="xs:string" />
       <xs:element minOccurs="0" name="Memory" type="xs:string" />
       <xs:element minOccurs="0" name="Displaysize" type="xs:string" />
       <xs:element minOccurs="0" name="Storage" type="xs:string" />
       <xs:element minOccurs="0" name="Costcenter" type="xs:string" />
       <xs:element minOccurs="0" name="Deviceprice" type="xs:string" />
       <xs:element minOccurs="0" name="Devicerent" type="xs:string" />
       <xs:element minOccurs="0" name="ContractNumber" type="xs:string" />
       <xs:element minOccurs="0" name="Length" type="xs:string" />
       <xs:element minOccurs="0" name="Startdate" type="xs:string" />
       <xs:element minOccurs="0" name="Enddate" type="xs:string" />
       <xs:element minOccurs="0" name="Signingdate" type="xs:string" />
```

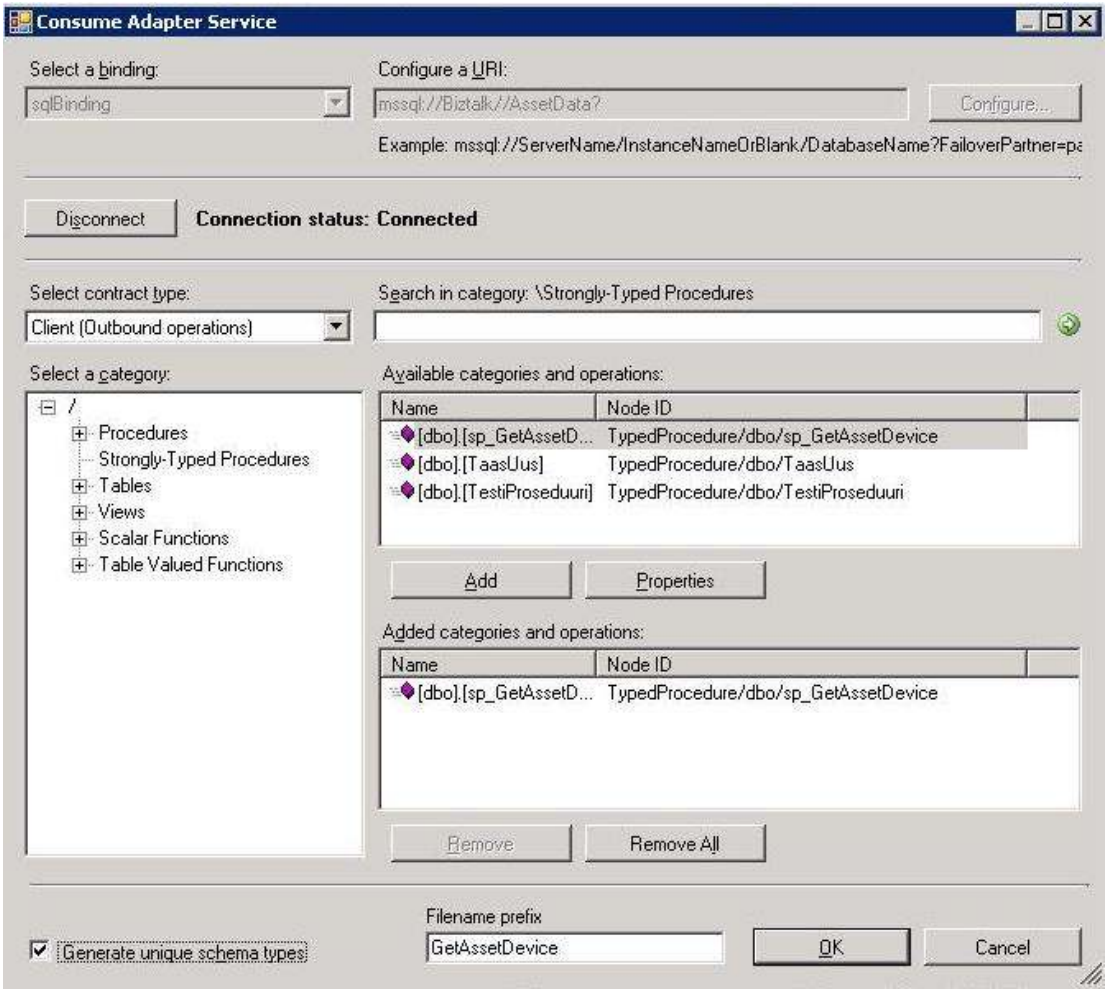
```
          <xs:element minOccurs="0" name="Contractprice" type="xs:string" />
          <xs:element minOccurs="0" name="ChangeDate" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Lastupdated">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="Date" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Tämä Schema sisältää oletuksena samoja nimiavaaruksia, kuin “SerialNumber”-Schemakin. ”targetNamespace”-nimiavaruus nimettiin Scheman nimen mukaan. Nimiavaruuden saa asetettua Scheman Properties-osiosta. Schemassa on kolme eri osiota. Ensimmäiseen tulevat SCCM-lähteestä saapuvat tiedot. Tämän elementin nimi on ”SMS_Answer”. Lapsielementteihin tulevat yksittäiset tiedot. Näihin elementteihin pitää määritellä ”MinOccurs”-arvoksi 0, koska muuten Schema vaatii tietoa elementteihin, vaikka tietolähteestä ei välttämättä löydy tarvittavaa tietoa. Muutoin orkestraatio voi ajautua virhetilanteeseen. Seuraava elementti on nimeltään ”Asset_Answer”. Tämän elementin alle tulevat lapsielementteinä AssetWebService-tietolähteen tiedot. Viimeinen elementti on nimeltään ”LastUpdated”. Tämän elementin lapsielementtinä toimii ”Date”, johon tulee Asset-tietokannan viimeisin päivitysaika. Elementissä ilmoitetaan päivämäärä sekä kellonaika. Scheman kaikki elementit ovat String-tyyppisiä, koska tiedoilla ei suoriteta minkäänlaista laskentaa.

5.5 Adapterin valinta ja liittäminen projektiin

Tässä opinnäytetyössä käytettiin WcfCustom-adapteria tietokantakyselyiden tekemiseen proseduurin avulla. Adapterin lisääminen projektiin löytyy Solution Ex-

plorerista: (Add Generated Items->Consume Adapter Service->Consume Adapter Service). Tämän jälkeen avautuu kuviossa 8 oleva ikkuna.



Kuvio 8. Adapterin luonti

Adapterin lisääminen tapahtuu samalla tavalla SCCM-tietokantaa varten. Ylhäältä vasemmalta valitaan ”Select Binding” -kohtaan ”sqlBinding”. Tämä määrittelee sidonnan automaattisesti generoidulle adapterille ja Schemoille. Seuraavaksi pitää määritellä yhteysmerkkijono. ”Configure”-nappia painikkeella avataan ikkuna, jonka ”URI properties” -välilehdellä voi määritellä Sql-palvelimen ja tietokannan nimen. ”Initial Catalog” on tietokannan nimi ja ”server” on Sql-palvelimen nimi. ”Connect”-painike muodostaa yhteyden tietokantaan. Seuraavaksi valitaan taulukon 2 mukaiset asetukset pääikkunassa.

Taulukko 2. Adapterin lisäysvaiheen asetukset

Asetus	**Määrittely**
Select Contract Type	Client (Outbound Operations)
Select a Category	Strongly-typed Procedures
Available categories and operations	sp_GetAssetDevice
Filename prefix	GetAssetDevice
Generate Unique Schema Types	X

5.5.1 Adapterin generoimat Schemat

Adapteri generoi kaksi Schemaa: ”GetAssetDeviceTypedProcedure.dbo.xsd” ja ”GetAssetDeviceProcedureResultSet.dbo.sp_GetAssetDevice.xsd”. Ensimmäinen Schema on ikään kuin äiti-Schema. Tämän Scheman avulla lähetetään parametri-sarjanumero Sql-palvelimelle. Toinen Schema sisällytetään äiti-Schemaan, kun vastaus saadaan tietolähteestä.

Liitteessä 5 kuvatulla Schemalla lähetetään proseduurikutsu Sql-palvelimelle. Parametri asetetaan elementtiin ”serialnumber”. Parametrin asetuksiin on määritetty, että se voi esiintyä kerran tai 0 kertaa. Parametrin tyypiksi on määritetty ”xs:string”. Elementti ”StoredProcedureResultSet0” vastaanottaa vastauksen.

Liitteessä 6 kuvattu Schema palauttaa kaikki tiedot GetAssetDeviceTypedProcedure.dbo-Schemalle. Elementit on määritelty ”xs:string”-tyyppisiksi. Niille on myös määritetty ”minOccurs”-arvoksi 0, koska elementtiin ei välttämättä tule tietoa, jos sitä ei löydy tietolähteestä, ”maxOccurs”-arvo on määritetty yhdeksi, koska tietona ei voi tulla kuin yksi tietokannan arvo. Schema on eräänlainen kokoelma palautetuista arvoista, jotka sisällytetään lopullisen Schemaan.

5.5.2 Automaattisesti generoitu asetustiedosto

Schemojen lisäksi generoidaan automaattisesti asetustiedosto fyysistä porttia varten. Asetustiedosto on Xml-tyyppinen tiedosto, jossa määritellään erilaisia asetuksia fyysistä porttia varten. Asetukset pitää aina tarkistaa portin luonnin jälkeen. Xml-tiedostossa on hyvä määritellä portille sitä kuvaava nimi ennen kuin siirtyy sen luomiseen. Nimi muokataan tiedostosta löytyvään ”SendPort”-elementin attribuutin ”Name”-arvoksi. Täydellinen kuvaus asetustiedostosta on liitteessä 7.

```
<SendPort Name="AssetSqlSend" IsStatic="true" IsTwoWay="true" BindingOp
tion="0">
```

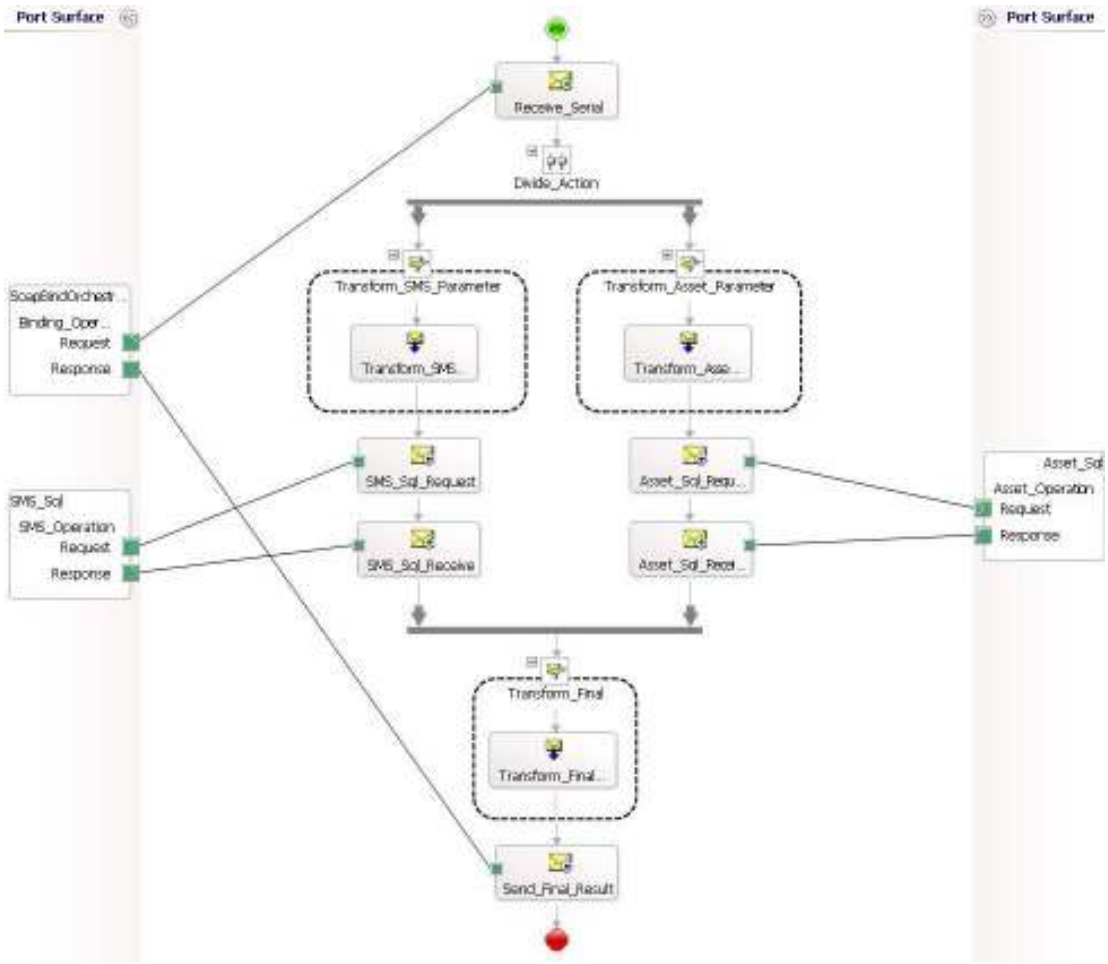
5.6 Orkestraatio

Tässä opinnäytetyössä luotiin orkestraatio, joka hakee kahdesta eri tietolähteestä parametrin avulla tietoa ja palauttaa ne kutsujalleen. Kuviossa 9 on opinnäytetyössä valmistunut orkestraatio.

Kuviossa vasemmalla on kaksi loogista porttia: ”SoapBindOrchestration” ja ”SMS_Sql”. Näistä ensimmäinen tuo viestin orkestraatioon ja toinen suorittaa proseduurikutsun SCCM-tietokantaan.

Kuviossa oikealla on looginen portti: ”Asset_Sql”. Tämä portti suorittaa proseduurikutsun Asset-tietokantaan.

Kuviossa keskellä on varsinainen prosessi, joka suoritetaan. Ensimmäisenä on Receive-elementti, joka vastaanottaa sarjanumeron. Tämän jälkeen orkestraatio jaetaan kahdeksi eri linjaksi. Molempien linjojen alkupuolella sarjanumero-parametri viedään proseduuria kutsuvalle elementille, jonka jälkeen tehdään kyselyt ja palautetaan vastaukset. Lopuksi ketjut yhdistetään ja tulokset kartoitetaan yhdelle Xml-dokumentille ja lähetetään vastaus kutsujalle.



Kuvio 9. Esimerkkiprojektin valmis orkestraatio

Orkestraation vaiheet tehdään seuraavassa järjestyksessä:

1. Elementtien asettaminen ja viestien luonti

2. Kartoituksien määrittely
3. Loogisten porttien määrittely
4. Orkestraation asetusten määrittely

5.6.1 Orkestraation perustaminen

Orkestraatio lisätään projektiin melkein samalla periaatteella kuin Schema. Projektin kohdalla Solution Explorerin pikavalikosta: (Add->New Item->Orchestration Files->Biztalk Orchestration). Tässä projektissa orkestraation nimeksi annettiin ”Orchestration_SMS_Asset.odx”. Orkestraation lisäämisen jälkeen avautuu orkestraatio-ikkuna, jolla tehdään lopullinen mallinnus.

5.6.2 Viestit ja elementit (Shapes)

Orkestraatioon on sijoitettu kuusi eri viestiä. Taulukossa 3 esitellään kaikki viestit ja niihin liittyvät Schemat.

Taulukko 3. Orkestraation elementit ja niiden selitykset esimerkkiprojektissa

Nimi	Tehtävä	Schema
Parameter	Tuoda parametri sisään orkestraatioon ja siirtää se seuraaville viesteille.	Serialnumber.xsd
SMS_Send	Viedä parametri Sql-palvelimelle.	GetSMSDeviceTypedProcedure.dbo.xsd

SMS_Receive	Vastaanottaa tulosjoukko Sql-palvelimelta.	GetSMSDeviceProcedureResultSet.dbo.xsd
Asset_Send	Viedä parametri Sql-palvelimelle.	GetAssetDeviceTypedProcedure.dbo.xsd
Asset_Receive	Vastaanottaa tulosjoukko Sql-palvelimelta.	GetAssetDeviceProcedureResultSet.dbo.xsd
Final_Result	Muodostaa lopullinen vastaus tulosjoukoista pyynnön tekijälle.	FinalAnswer.xsd

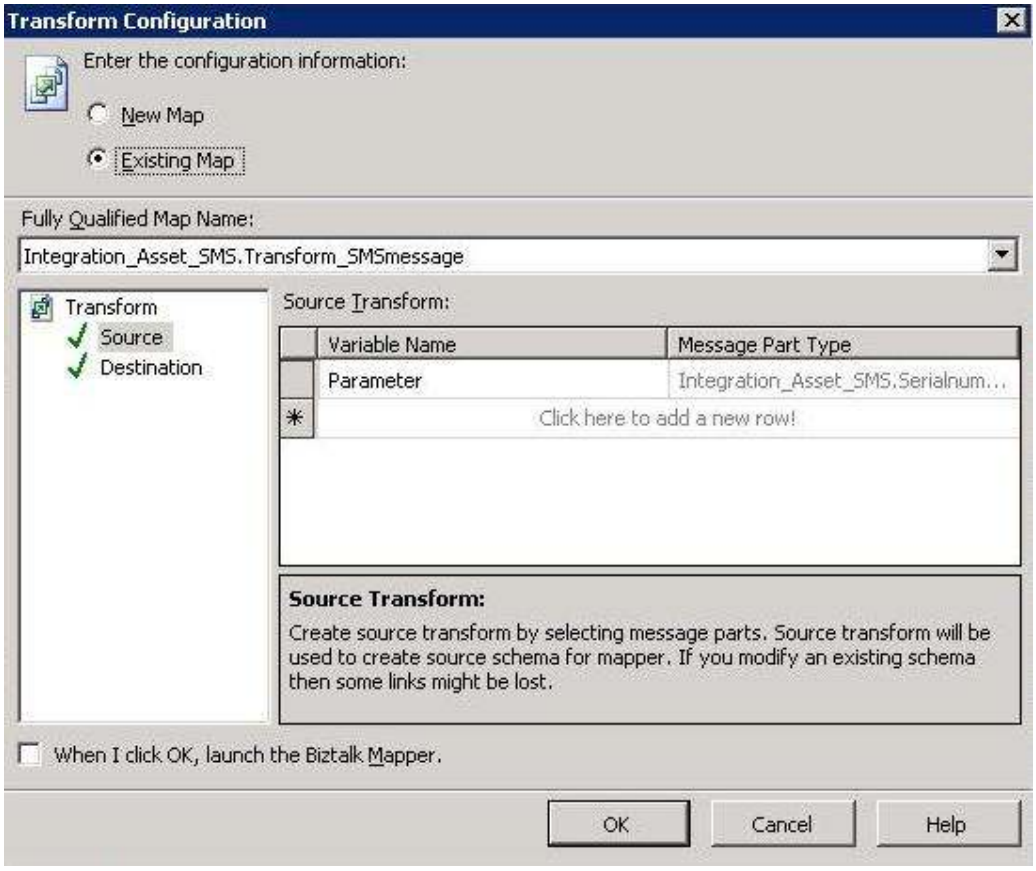
Viestit liitetään Biztalkissa elementteihin. Orkestraatiossa on käytetty seuraavia elementtejä: Receive, Send, Parallel Actions, Transform ja Construct. Ensimmäinen orkestraatiossa esiintyvä elementti on Receive. Tähän elementtiin liitetään ”Parameter”-viesti. Viesti tuo mukanaan sarjanumero parametrin orkestraatioon. Seuraavaksi orkestraatio jakautuu kahtia Parallel Actions -elementin avulla. Toinen linja vie viestin Asset-tietokannalle ja toinen SCCM-tietokannalle. Esimerkiksi Asset-tietokannan tapauksessa viesti ”Parameter” kartoitetaan ”Asset_Send”-viestille. ”Asset_Send” suorittaa kyselyn tietokantaan ja sieltä palautetaan tulosjoukko ”Asset_Receive”-viestin mukana. Viestin kulku toimii samalla tavalla SCCM-tietokannan tapauksessa. ”Asset_Receive”- ja ”SMS_Receive” -viestit kartoitetaan lopuksi yhdeksi kokonaisuudeksi eli ”Final_Result”-viestiksi. ”Final_Result”-viesti lähetetään takaisin kutsujalle. Orkestraatiossa ei tarvinnut käyttää muita elementtejä, koska kyseessä oli Request-Response-operaatio. Kysyjä haluaa vastauksen ja se lähetetään hänelle, jos vastaus löytyy. Orkestraatiossa ei

ollut tarvetta esimerkiksi erilaisille toistoille tai ulkopuolisten orkestraatioiden käynnistyksille.

5.6.3 Kartoitukset (Mappings)

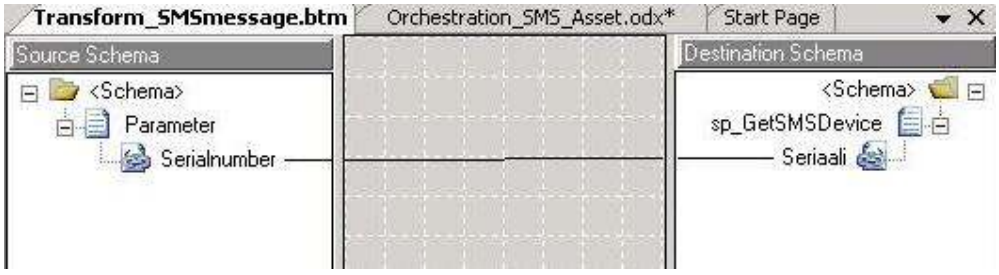
Kartoitukset tarvitsevat Construct- ja Transform-elementit viestin konvertointiin. Tässä orkestraatiossa kartoituksia tarvittiin kolmessa eri paikassa. Parametri piti kartoittaa tietokantaoperaatioita varten ja lopuksi kartoitettiin lopullinen vastausviesti. Orkestraatiossa Construct-elementti sijoitetaan ensin ja sen jälkeen sen sisään sijoitetaan Transform-elementti. Construct-elementin ominaisuuksiin pitää määritellä, minkä viestin haluaa tuottaa lopputuloksena sekä elementin nimi. Transform-elementti vaatii ominaisuusikkunassa sisään tulevat viestit tai viestin parametriksi.

Kuviossa 10 esitellään parametrin kartoitus SCCM-tietokantaoperaatiolle. Sisään tulevia ja lähteviä viestejä voi olla enemmänkin kuin yksi. Tämä antaa paljon joustavuutta erilaisille kartoituksille. Sisään tuleva viesti on ”Parameter”. Kartoitukseen jälkeen valmistuva viesti on ”SMS_Send”. Vasemmassa alakulmassa on valinta: ”When I click OK, launch the Biztalk Mapper”, tämä käynnistää kartoitusmanagerin. Seuraavassa kartoitetaan parametri SCCM-tietokantaoperaatiota varten.



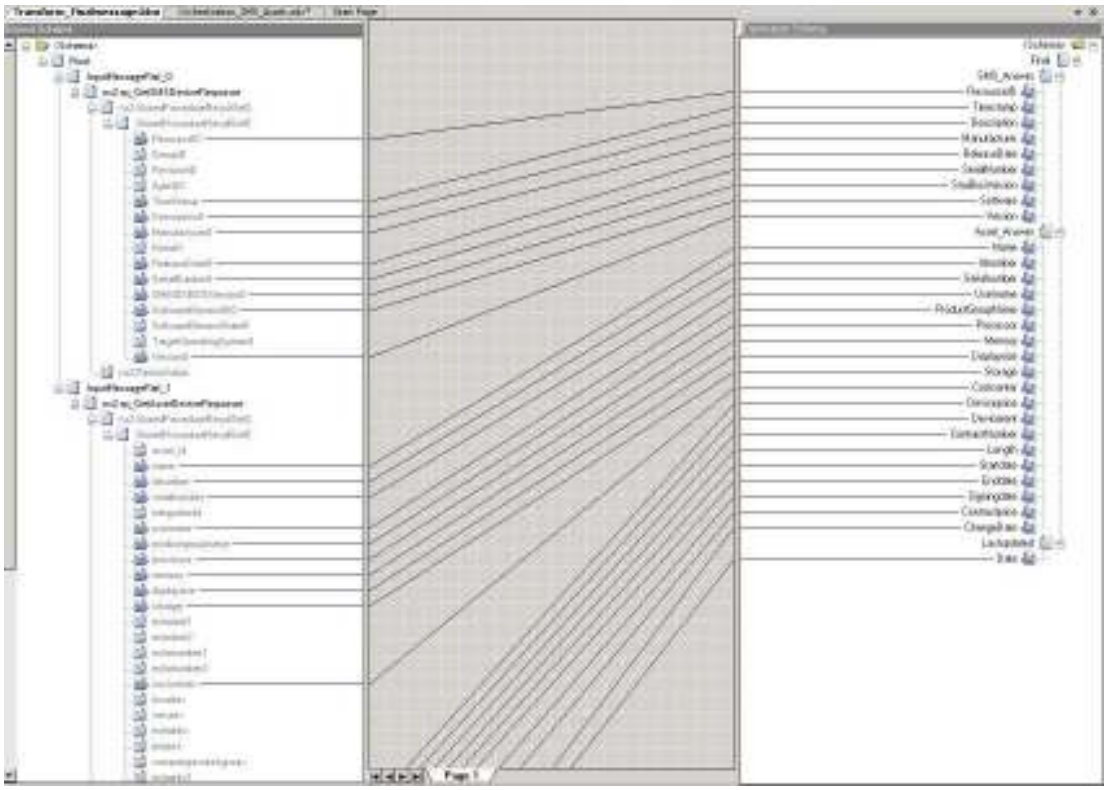
Kuvio 10. Kartoituksen konfigurointi

Kuviossa 10 on Biztalkin kartoitusmanagerin näkymä siitä kuinka sarjanumero-parametri viedään vastaanottavalta elementiltä tietokantaoperaatioita suorittaville elementeille. Kuviossa 11 on esimerkkinä SCCM-tietokannan parametrin vienti. Asset-tietokannan kohdalla parametrin vienti tapahtuu samalla tavalla.



Kuvio 11. Parametrin kartoittaminen

Kuviossa 12 on Biztalkin kartoitusmanagerin näkymä siitä kuinka molempien tietokantaoperaatioiden palauttamat tiedot viedään ”FinalAnswer.xsd”-Scheman määrittelemälle Xml-dokumentille.



Kuvio 12. Esimerkkiprojektin loppukartoitus

5.6.4 Loogiset portit

Orkestraatiota varten luotiin kolme loogista porttia. Portit yhdistetään elementteihin niiden loogisen toiminnan mukaan. Yhdistämisen pystyy tekemään ”raahaamalla” viivan portista elementtiin. Ensimmäinen looginen portti on nimeltään ”SoapBindOrchestration”. Tämän portin kommunikaatiokaava on request-response. Portti tekee Webservice-kutsun orkestraatiolle, joten sen rajoituksiin piti määritellä ”Public”. Kommunikaatiosuunnaksi määriteltiin ”I’ll be receiving a request and sending a response”. “Port Binding” -kohtaan määriteltiin “Specify Later”.

Seuraava portti on nimeltään “Sms_Sql”. Tämän portin tehtävänä on suorittaa Sql-operaatio SCCM-tietokantaan. Kommunikaatiokaavaksi määriteltiin request-response. Tämä portti suorittaa palvelimella Sql-kutsun, joten sen rajoituksiin voitiin määritellä ”Internal”. Kommunikaatiosuunnaksi määriteltiin ”I’ll be receiving a request and sending a response”. “Port Binding”-kohtaan määriteltiin “Specify Later”. Toinen samanlainen portti luotiin Asset-tietokantaa varten. Sen nimeksi annettiin ”Asset_Sql”.

Loogiset portit, kun orkestraatiota testataan tiedostolla

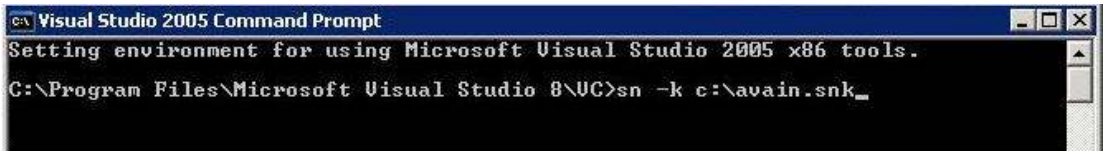
Orkestraation testaaminen tiedostolla on järkevää, koska muuten ei voi tietää toimiiko koko projekti käytännössä. Orkestraatiota ei kannata julkaista Webserviceksi, jos projekti ei toimi ja siinä on virheitä. Tiedostolla testattaessa orkestraatioon pitää luoda kaksi erinäistä loogista porttia yhden Webservice-portin sijaan. Portteja luodaan kaksi, koska tiedostojärjestelmään tehdään kaksi eri hakemistoa. Toinen vastaanottaa viestin ja toiseen palautetaan valmis vastaus. Webservicen kohdalla ei tarvita kahta eri porttia, koska se toimii ”lähetä ja vastaanota”-periaatteella.

Ensimmäinen portti määritellään tuomaan viesti sisään. Tämän portin nimi voisi olla ”File_In”. Portin kommunikaatiokaavaksi pitäisi määrittää ”One-Way”. Rajoituksiin voi laittaa ”Internal”, koska tiedoston voi pudottaa järjestelmään sen sisäpuolella. Kommunikaatiosuunnaksi määriteltäisiin ”I’ll always be receiving messages on this port”.

Toinen portti määritellään lähettämään viesti pois orkestraatiosta. Tämän portin nimi voisi olla ”File_Out”. Kommunikaatiosuunnaksi määriteltäisiin ”I’ll always be sending messages on this port”.

5.6.5 Orkestraation asetukset

Ennen orkestraation kääntämistä projektille pitää luoda ”Strong Named Key”. Tämä avain yksilöi eri projektit toisistaan. Avain luodaan Visual Studion Command Prompt -konsolissa kuviossa 13 esitetyllä tavalla.



Kuvio 13. Strong named key -avaimen luonti

Avaimen luonnin jälkeen se pitää kirjata projektin kokoonpanoasetuksiin Solution Explorerista valitsemalla ”Properties” -kohta pikavalikosta. ”Assembly”-välilehdeltä löytyy kohta ”Assembly Key File”. Tähän kohtaan määritellään avaimen sijaintipolku.

Orkestraatio pitää ensin kääntää. Tämä tapahtuu Solution Explorerissa ja valitsemalla ”Build” pikavalikosta. Yleisin virhe kääntämisen aikana on: ”you must specify at least one already-initialized correlation set for a non-activation receive that is on a non-selfcorrelating port”. Ensimmäisen Receive-elementin ominaisuuksista pitää asettaa “Activate”-arvoksi “True”. Kääntämisen jälkeen suoritetaan “Deploy” eli varsinainen projektin julkaisu. Tämän jälkeen voidaan siirtyä määrittelemään fyysisiä portteja, jos virheitä ei ilmaantunut.

5.7 Fyysiset portit ja niiden luonti asetustiedoston avulla

Fyysisiä portteja luotiin tässä projektissa kolme kappaletta. Ensimmäinen tuo viestin sisään Biztalkiin ja lähettää vastauksen kutsujalle. Tämän portin luominen käsitellään myöhemmin Webservicen rakentamisen yhteydessä. Kaksi muuta toimivat tietokantaoperaatioiden suorittajina. Tietokantaoperaatioihin osallistuvien porttien

nimeksi tulivat ”AssetSqlSend” ja ”SMSSqlSend”. Portit ovat molemmat lähettäviä portteja.

Esimerkiksi ”AssetSqlSend”-portin luonti onnistuu seuraavanlaisesti. Otetaan asetustiedosto, joka on luotu adapterin luonnin yhteydessä, ja muokataan portille uusi nimi. Nimi on helppo muokata tiedostosta löytyvään ”SendPort”-elementin attribuutin ”Name”-arvoksi.

Seuraavaksi suoritetaan portin luonnin neljä vaihetta:

1. Avataan Biztalk Server Administration Console.

2. Konsolin Application-kohdan alta haetaan se projekti, johon porttia ollaan lisäämässä. Projektin kohdalla avataan pikavalikko ja valitaan: (Import->Bindings..)

3. Haetaan muokattu portin asetustiedosto ja avataan se.

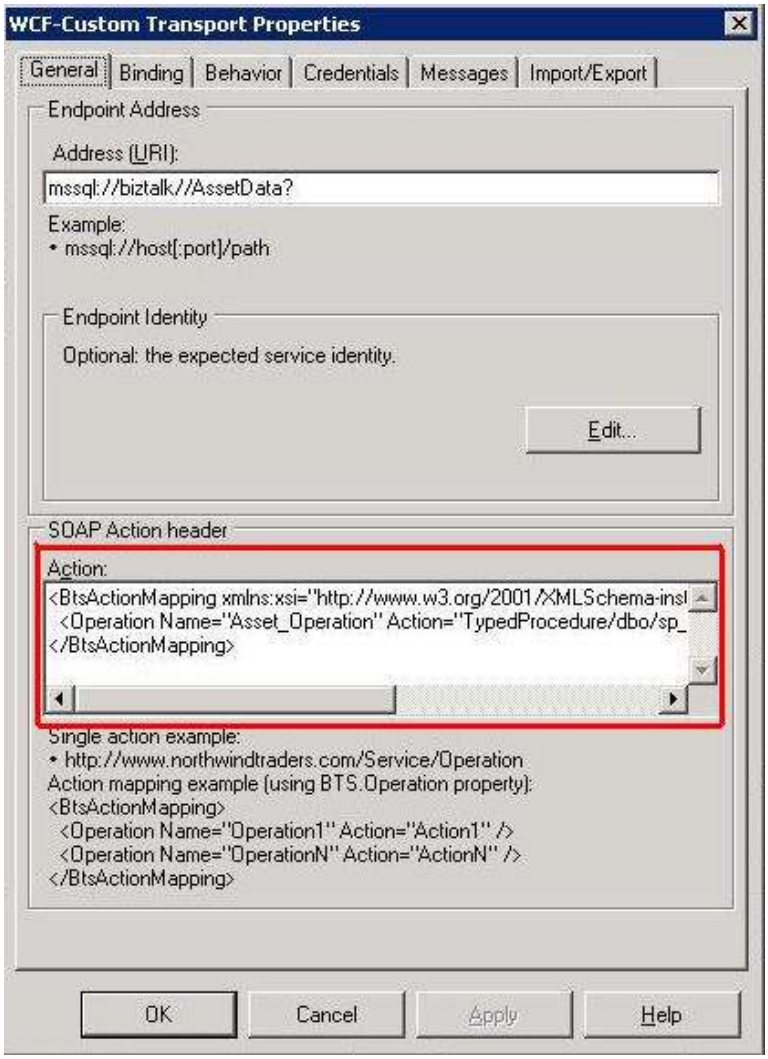
4. Portin luomisen onnistumisen voi tarkistaa Send Ports -kohdasta.

Porttia ei kannata vielä tässä vaiheessa käynnistää. Opinnäytetyössä porttien asetusten muokkaus tehtiin usein Visual Studion kautta. Portin löytää Visual Studiossa Biztalk Explorerista. Send Ports -kohdasta voi aukaista näkyviin kaikki lähettävät portit.

Seuraavaksi avataan portin konfiguraationäkymä pikavalikosta valitsemalla ”Edit”. Tämän jälkeen valitaan Address(Uri) -kohta. Tästä avautuu erilaisia ominaisuuksia portin konfigurointiin. Kuvion 14 konfigurointi-ikkunalla määritetään keskeiset ominaisuudet portille.

Kuvion 14 SOAP Action header -kohtaa pitää muuttaa, niin että ”Operation”-elementin attribuutin ”Name” arvoksi tulee sen operaation nimi, mikä on määritelty

loogiselle portille. Tämän nimen voi tarkastaa orkestraatiossa loogisen portin ominaisuuksista. Yleensä Biztalk lisää oletuksena portin operaatioksi ”Operation_1” jne. Operaatioiden nimet kannattavat olla yleensä toisistaan eroavia. Silloin kun operaation nimeä muokataan orkestraatiossa, niin kannattaa ensin pysäyttää kaikki käynnissä olevat instanssit ja sen jälkeen käynnistää vielä uudelleen ”Host Instance”. Luonnollisesti projekti pitää kääntää uudelleen ja käynnistää ”Deploy”, jos siihen tehdään tämän kaltaisia muutoksia. Seuraavaksi kannattaa vielä tarkistaa ”Send”-välilehdeltä, että pipelinet on määritelty oikein. Lähettävänä pipelinena pitäisi olla XmlTransmit ja vastaanottavana XmlReceive. Nämä pipelinet tarvitaan, koska muuten viestejä ei saisi kartoitettua orkestraatiossa.



Kuvio 14. Fyysisen tietokantaportin konfigurointi

5.7.1 Fyysiset portit, kun orkestraatiota testataan tiedostolla

Orkestraation testaaminen tiedostolla edellyttää fyysisille porteille niitä vastaavat loogiset portit. Fyysisistä porteista ensin luodaan vastaanottava fyysinen portti. Portti luodaan valitsemalla Visual Studion Biztalk Explorerissa pikavalikosta: ”Add Receive Port”. Tämän jälkeen suoritetaan seuraavat kahdeksan vaihetta:

1. Kommunikaatiosuunnaksi valitaan ”One-Way”

2. Annetaan portille nimi ja tallenetaan valitsemalla ”Ok”

3. Asetetaan ”Receive location”. Tämä tapahtuu valitsemalla ”Receive location” -kohdan pikavalikosta: ”Add receive location”

4. ”Transport-type” -kohtaan valitaan ”File”

5. ”Address(Uri)” -kohdasta saa avattua lisää konfigurointimahdollisuuksia. Täältä valitaan polku tiedostojärjestelmässä, mihin sisään tulevan viestin tiedosto pudotetaan.

6. Pipelineksi valitaan XmlReceive.

7. Luodaan testitiedosto. Solution Explorerissa valitaan sisään tulevan viestin Scheman pikavalikosta: ”Properties”. ”Output Instance Filename”-kohdasta saa uuden konfiguraatioikkunan näkyviin, josta voi määritellä mihin testitiedosto luodaan.

8. Valitaan Schema-pikavalikosta: ”Generate Instance” ja testitiedosto generoituu tiedostojärjestelmään.

Seuraavaksi luodaan fyysinen portti, joka lähettää vastauksen tiedostojärjestelmään. Luonti koostuu kuudesta eri vaiheesta:

1. ”Send Ports” -kohdasta valitaan pikavalikosta: ”Add Send Port”.

2. Kommunikaatiosuunnaksi valitaan ”Static One-Way Port”.

3. ”Transport-type”-kohtaan valitaan ”File”.

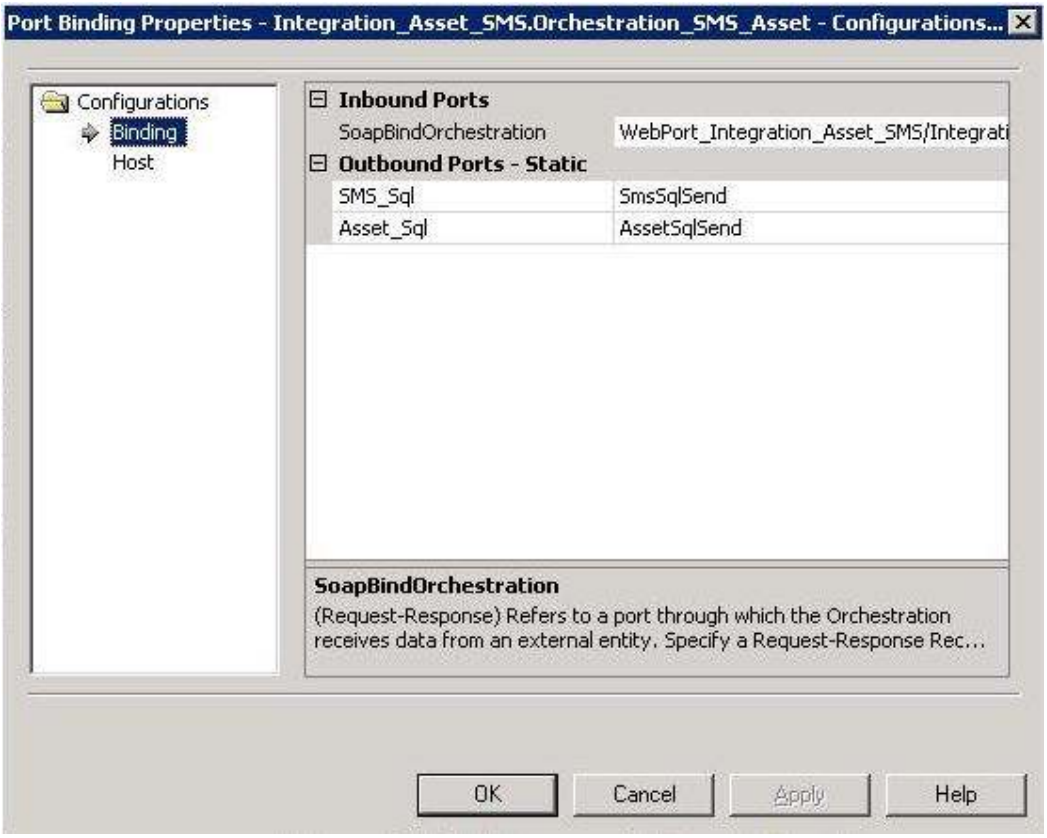
4. Annetaan portille nimi.

5. ”Address(Uri)”-kohtaan määritellään tiedostopolku, mihin valmiit vastaustiedostot generoidaan.

6. Valitaan pipelineksi XmlTransmit ”Send”-välilehdeltä.

5.7.2 Fyysisten porttien sitominen orkestraation loogisiin portteihin

Ennen kuin projekti on lopullisesti valmis käynnistettäväksi, niin fyysiset portit pitää sitoa vastaaviin loogisiin portteihin. Esimerkiksi Visual Studion Biztalk Explorerissa se käy seuraavanlaisesti. Valitaan orkestraation pikavalikosta: (Bind..). Kuviossa 15 on opinnäytetyössä käytetty sidonta.



Kuvio 15. Orkestraation sitominen fyysisiin portteihin

Kun loogisille porteille on löydetty toisiaan vastaavat fyysiset portit, niin ”Host”-välilehdeltä pitää vielä valita ”Host”-kohtaan ”BizTalkServerApplication”. Tämän jälkeen orkestraatio on valmis käynnistettäväksi.

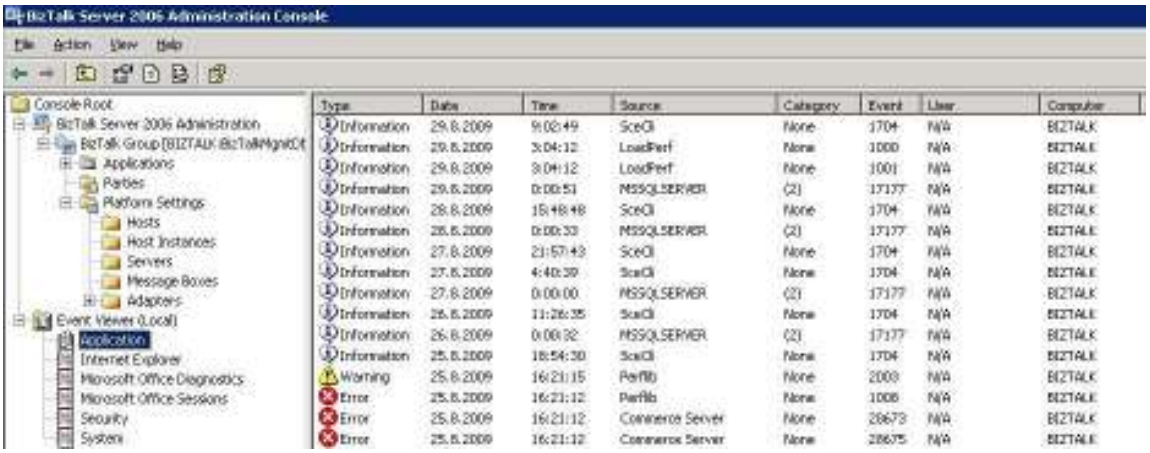
5.8 Biztalkin hallintakonsolin käyttö

Biztalkin hallintakonsoli on hyvä paikka eri Biztalk-sovellusten hallintaan. Sieltä pystyy hallitsemaan orkestraation, porttien, Schemojen ja monia muita konfiguraatioasetuksia. Hallintakonsolissa on myös kattava raportointijärjestelmä virheiden varalle. Viestin kulkua pystyy esimerkiksi seuraamaan askel askeleelta.

Esimerkiksi kuviossa 16 on Biztalk-hallintakonsolin (Event Viewer->Application)-välilehti. Virheen kohdalla voi avata pikavalikon, josta valitaan: ”Properties”. Täältä

löytyy yleensä virheen syy, jota voidaan lähteä selvittämään, esimerkiksi Googlen avulla.

Kuviossa 16 näkyy myös vasemmassa laidassa muita tärkeitä konfiguraatio mahdollisuuksia. Esimerkiksi ”Host Instances” -kohdasta saa käynnistettyä Host-instancen uudelleen. Tämä toimenpide kannattaa suorittaa aina, kun tekee projektiin muutoksia.



Kuvio 16. Biztalk-hallintakonsoli

5.9 Webservice

Opinnäytetyön lopuksi tehtävä Webservice oli luonteva jatko Biztalk-projektille, koska Webserviceä voitaisiin käyttää mistä tahansa järjestelmästä, joka tukee SOAP-protokollaa. Näin Webservice toimii käytännöllisenä rajapintana Biztalk-sovellukselle.

5.9.1 Webservicen luonti apuohjelmaa käyttäen

Biztalkin asennuksen mukana tulee apuohjelma, jolla pystyy luomaan Webservice -kokonaisuuden orkestraation avulla vaihe kerrallaan. Apuohjelman nimi on: ”BizTalk Web Services Publishing Wizard”. Webservice julkaistaan IIS-palvelimelle,

josta sitä voi myöhemmin kutsua asiakasohjelmalla. Ohjelma voidaan käynnistää seuraavasta paikasta: Start->Programs->Microsoft Biztalk Server 2006-> BizTalk Web Services Publishing Wizard. Kuviossa 17 on ohjelman aloitusikkuna.

Kuvio 17. Webservices publishing wizard

Webservicen luonti jakautuu ikkunoittain kuuteen vaiheeseen:

1. Valitaan Webservicen luonti orkestraation perusteella.

2. Asetetaan projektin kokoonpanotiedosto. Tämä tiedosto löytyy yleensä projektin kansiosta: (bin->Development). Tiedosto on dll-päätteinen kirjasto.

3. Arvioidaan ikkunassa olevan raportin perusteella, mitä orkestraatiosta voidaan viedä Webservicelle. Yleensä tämä tarkoittaa orkestraatioita ja loogisia portteja. Tällä kertaa valittiin kaikki mahdolliset orkestraation komponentit avuksi Webservicen luontiin. Tässä ikkunassa on myös mahdollista valita kaikki mahdolliset loogiset portit ja yhdistää ne yhdeksi Webserviceksi.

Tämä tapahtuu kohdasta:”merge all selected ports into a single webservice”.

4. Valitaan perustettavalle Webservicelle oma yksilöllinen nimiavaruus. Nimiavaruudeksi valittiin: “ http://Asset_Sms_IntegrationService”. Nimiavaruudeksi voi valita minkä tahansa Url-osoitteen. Yleensä testausvaiheessa käytetään tempuri.org-alkuista osoitetta. Tämä on yleisesti tunnettu nimiavaruus Asp.Net Webservice -instanssien testaukseen. Ikkunassa kysytään myös kolme muuta kysymystä. Nämä kysymykset liittyvät SOAP-protokollan yksityiskohtaisiin ominaisuuksiin. Tässä opinnäytetyössä ei käytetty näitä ominaisuuksia, joten ne voitiin jättää huomioimatta.

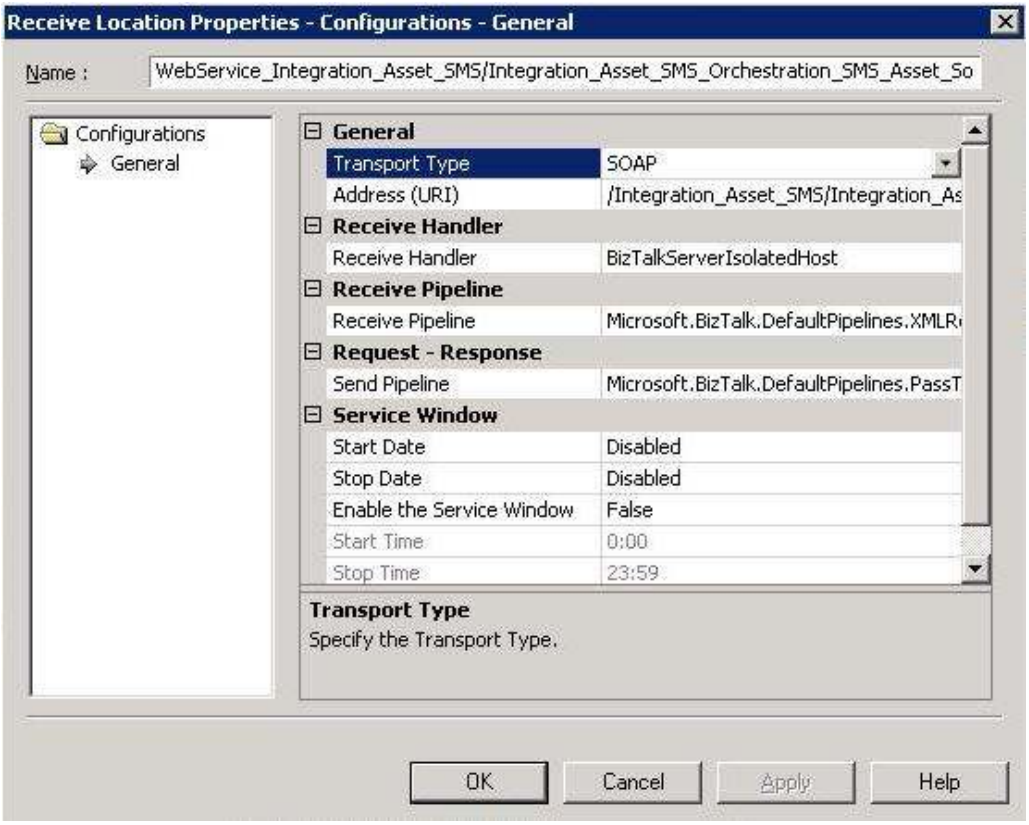
5. Määritellään Webservicelle sen paikka IIS-palvelimella. Osoite on testausvaiheessa yleensä muodossa: ”localhost”. Tässä projektissa Webservice luotiin osoitteeseen: ”http://localhost/Integration_Asset_SMS/”. Ikkunassa on myös mahdollista valita sellainen vaihtoehto, joka luo automaattisesti haluttuun Biztalk-sovellukseen portin, jota kautta tietojenvaihto Webservicelle onnistuu. Tässä kohdassa valittiin halutuksi sovellukseksi: ”SMS_Asset_Integration”.

6. Lopuksi apuohjelma luo raportin, josta voi tarkistaa tulivatko kaikki tiedot oikein ennen Webservicen generointia palvelimelle. Webservicen generointi lähtee käyntiin, kun painetaan ”Create”. Generoinnin jälkeen Webservice löytyy IIS-palvelimelta. (How to Use the BizTalk Web Services Publishing Wizard to Publish an Orchestration as a Web Service. 2009.)

5.9.2 Fyysinen SOAP-portti

Webservicen luontiin tarkoitettu apuohjelma luo fyysisen portin haluttuun sovellukseen. Tässä opinnäytetyössä portti luotiin sovellukseen: ”Integration_Asset_SMS”. Portin nimeksi generoitui: ”Web-

Port_Integration_Asset_SMS/Integration_Asset_SMS_Orchestration_SMS_Asset_SoapBindOrchestration”. Webservicen luonnin yhteydessä generoitavat fyysiset portit ovat poikkeuksetta vastaanottavia portteja. Kuviossa 18 on portin vastaanottopaikan konfiguraatio.



Kuvio 18. Fyysinen SOAP-portti

Taulukossa 4 on esitelty fyysisen SOAP-portin vaatimat asetukset. Biztalk generoi tämän portin valmiiksi, mutta siltä saattaa silti puuttua tärkeitä asetuksia. Taulukon 4 asetuksista jouduttiin muokkaamaan Receive Pipelineksi XmlReceive. Pipeline oli alun perin PassThruReceive. Opinnäytetyössä käytetty orkestraatio pystyy ottamaan vastaan viestin ainoastaan XmlReceive-pipelinen kautta.

Taulukko 4. Fyysisen SOAP-portin asetukset

Asetus	**Määritys**

Transport Type	SOAP
Address(Uri)	/Integration_Asset_SMS/…
Receive Handler	BiztalkServerIsolatedHost
Receive Pipeline	XmlReceive
Send Pipeline	PassThruTransmit

5.9.3 Webservicen vienti IIS-palvelimelle

Webservicen perustaminen IIS (Internet Information Services) -palvelimelle onnistuu avaamalla IIS-manager. Managerista pitää luoda uusi ”Website”. Se tapahtuu valitsemalla pikavalikosta: (Websites->New->Website). Tämän jälkeen avautuu ohjattu web-sivun luominen palvelimelle.

1. Web-sivun kuvaukseksi voidaan laittaa mikä tahansa sitä kuvaava nimi.

2. Määritellään TCP-portti web-sivulle. Tälle portille määritellään useimmiten arvo: ”80”. Tässä projektissa kuitenkin portin arvoksi tuli ”83”. Portin arvo ei muuta toiminnallisuutta millään tavalla.

3. Haetaan sivuston sijainti. Tämä sijainti on yleensä: ”C:\Inetpub\wwwroot”. Tässä ikkunassa on myös mahdollisuus antaa kaikille osapuolille oikeus käyttää sivustoa.

4. Valitaan sivustolle viidestä eri oikeudesta sopivat. Oikeuksia ovat: luku, skriptaus, suoritus, kirjoitus ja selailuoikeudet. Tässä projektissa valittiin kaikki oikeudet käyttöön.

5. Loppuvaiheessa web-sivun luonnin jälkeen pitää tehdä muutama muutos sen asetuksiin. Avataan luodun web-sivun pikavalikko ja valitaan ”Properties”. Tämän jälkeen valitaan ”Asp.Net”-välilehti. Täältä pitää muuttaa Asp.Netin versioksi 2.0.50727. Tämä on tärkeä muutos, koska muuten Webservice ei tule toimimaan. Web-sivu pitää myös muistaa käynnistää, jotta se olisi saatavilla. Tämän jälkeen Webservice pitäisi löytyä luodusta osoitteesta, joka on muotoa: ”http://localhost:80/Webservice”.

5.9.4 Webservicen asiakasohjelma

Webservicen asiakasohjelma kehitettiin Asp.Net-palvelinohjelmointikielellä. Web-sivu luodaan Visual Studiossa valitsemalla: (New Website->Asp.Net Website). Webservicen kutsuminen Asp.Net-ohjelmassa aloitetaan lisäämällä se referenssiksi Asp.Net-projektiin. Se tapahtuu Solution Explorerin pikavalikosta valitsemalla: ”Add Web Reference”. Tämä toiminto avaa ikkunan, johon pitää kirjoittaa Webservicen Url-osoite. Toiminnon pitäisi tuoda ikkunaan saatavilla olevat Webservicet linkkeinä. Sitten kun haluttu Webservice on löydetty, niin se lisätään projektiin referenssiksi valitsemalla ”Add Reference”.

Esimerkki 12. Asp.Net Codebehind WsClient.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
```

```
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void hae_Click(object sender, EventArgs e)
    {
        //Luodaan uusi parametriobjekti
        Asset_SMS_WS.Parameter Param = new Asset_SMS_WS.Parameter();
        //Asetetaan parametri TextBoxista objektin ominaisuudeksi
        Param.Serialnumber = Hakuparametri.Text;

        //Luodaan Webserviceobjekti

Asset_SMS_WS.Integration_Asset_SMS_Orchestration_SMS_Asset_SoapBindOrches
tration ws = new
Asset_SMS_WS.Integration_Asset_SMS_Orchestration_SMS_Asset_SoapBindOrches
tration();
        //Kutsutaan Webserviceä parametrin avulla
        Asset_SMS_WS.Final final = ws.Binding_Operation(Param);

        //Jos SCCM-lähde ei ole tyhjä
        if (final.SMS_Answer != null)
        {
            //Näytetään sen taulukko
            resultset1.Attributes.Add("style", "display:inline;");
            virhe.Text = "";
            //Lisätään vastaukset taulukkoon
            valmistaja.Text = final.SMS_Answer.Manufacturer;
            kuvaus.Text = final.SMS_Answer.Description;
            julkaisu.Text = final.SMS_Answer.ReleaseDate;
            smsid.Text = final.SMS_Answer.ResourceID;
            sarjanro.Text = final.SMS_Answer.SerialNumber;
            biosversion.Text = final.SMS_Answer.SmsBiosVersion;
            ohjelmisto.Text = final.SMS_Answer.Software;
            aika.Text = final.SMS_Answer.Timestamp;
            smsversio.Text = final.SMS_Answer.Version;
        }
        //Jos Asset-lähde ei ole tyhjä
        if (final.Asset_Answer != null)
        {
            //Näytetään sen taulukko
            resultset2.Attributes.Add("style", "display:inline;");
            virhe.Text = "";
            //Lisätään vastaukset taulukkoon
            muutos.Text = final.Asset_Answer.ChangeDate;
            sopimus.Text = final.Asset_Answer.ContractNumber;
            shinta.Text = final.Asset_Answer.Contractprice;
            kustannus.Text = final.Asset_Answer.Costcenter;
            laitehinta.Text = final.Asset_Answer.Deviceprice;
            laitevuokra.Text = final.Asset_Answer.Devicerent;
            naytto.Text = final.Asset_Answer.Displaysize;
            loppupvm.Text = final.Asset_Answer.Enddate;
            assetid.Text = final.Asset_Answer.Idnumber;
            pituus.Text = final.Asset_Answer.Length;
            muisti.Text = final.Asset_Answer.Memory;
            nimi.Text = final.Asset_Answer.Name;
            prosessori.Text = final.Asset_Answer.Processor;
            laiteryh.Text = final.Asset_Answer.ProductGroupName;
```

```
            asarjanro.Text = final.Asset_Answer.Serialnumber;
            allekirjoitus.Text = final.Asset_Answer.Signingdate;
            alkupaiva.Text = final.Asset_Answer.Startdate;
            varasto.Text = final.Asset_Answer.Storage;
            kayttaja.Text = final.Asset_Answer.Username;

            paivitys.Text = final.Lastupdated.Date;
        }
        //Jos SCCM-lähde on tyhjä
        if (final.SMS_Answer == null)
        {
            //Piilotetaan taulukko
            resultset1.Attributes.Add("style", "display:none");
            //Laitetaan tyhjät arvot taulukkoon
            valmistaja.Text = "";
            kuvaus.Text = "";
            julkaisu.Text = "";
            smsid.Text = "";
            sarjanro.Text = "";
            biosversion.Text = "";
            ohjelmisto.Text = "";
            aika.Text = "";
            smsversio.Text = "";
        }
        //Jos Asset-lähde on tyhjä
        if (final.Asset_Answer == null)
        {
            //Piilotetaan taulukko
            resultset2.Attributes.Add("style", "display:none");
            //Laitetaan tyhjät arvot taulukkoon
            muutos.Text = "";
            sopimus.Text = "";
            shinta.Text = "";
            kustannus.Text = "";
            laitehinta.Text = "";
            laitevuokra.Text = "";
            naytto.Text = "";
            loppupvm.Text = "";
            assetid.Text = "";
            pituus.Text = "";
            muisti.Text = "";
            nimi.Text = "";
            prosessori.Text = "";
            laiteryh.Text = "";
            asarjanro.Text = "";
            allekirjoitus.Text = "";
            alkupaiva.Text = "";
            varasto.Text = "";
            kayttaja.Text = "";
            paivitys.Text = "";
        }
        //Jos kummastakaan lähteestä ei löydy tietoja
        if (final.SMS_Answer == null && final.Asset_Answer == null)
        {
            //..niin annetaan virheilmoitus
            virhe.Text = "Ei hakutuloksia";
        }

    }
```

```
}
```

Yllä kuvatussa ohjelmassa luodaan Webservice-objekti sekä parametri-objekti. Parametri-objektille asetetaan parametri, joka asetetaan Webservice-objektille ominaisuudeksi. Lopuksi kutsutaan Webserviceä parametrina avulla ja asetetaan tiedot käyttöliittymän puolelle. Liitteessä 8 on kuvattu käyttöliittymän ohjelmointipuoli. Liitteessä 9 on kuvankaappaus lopullisesta asiakasohjelmasta.

6 JATKOKEHITYS

Jatkokehityksen mahdollisuudet ovat hyvät, koska palvelimelle asetettua Webserviceä voi kutsua mistä tahansa järjestelmästä. Webserviceä voisi kutsua esimerkiksi seuraavassa Biztalk-projektissa osana orkestraatiota. Atk-henkilöstön toiveena oli, että sovellus voisi toimia joskus toiseenkiin suuntaan eli päivityksiin, poistoihin ja lisäyksiin.

SCCM-tietokannan kohdalla muutosten teko on jo nyt mahdollista. Tietokannan eheyden kannalta on kuitenkin kannattavaa tutkia ensin sen kriittiset osat. Käytössä olevien tietokantojen muokkaamisessa on aina omat vaaransa. AssetWebServicen kohdalla päivitysten tekeminen voisi olla mahdollista julkaisemalla oma Webservice, joka perustuu dynaamiseen ohjelmaan. Tätä Webserviceä voitaisiin kutsua Biztalkissa. Ohjelman tekemisessä on oikeastaan vain yksi ongelma. AssetWebservicen SSL-sertifikaatti ei ole validi. Tämä aiheuttaa palvelun hidastumisen, koska salaus pitää kiertää ohjelmallisesti.

Mielenkiintoisiin kehityskohteisiin kuuluu Seinäjoen ammattikorkeakoulussa toimiva ”Helppari”-järjestelmä, josta saa apua tietoteknisiin ongelmiin. Helppari toimii Microsoft Sharepoint -järjestelmän päällä. Biztalk tukee hyvin Sharepoint-integraatiota. Tällä tavalla saataisiin yhdistettyä monenlaisia tietoja laitteesta esimerkiksi vikailmoitukseen.

Biztalk Serveriä kannattaa käyttää vain sellaisissa tilanteissa, missä se on järkevää. Sitä ei kannata käyttää jokaiseen ohjelmistoprojektiin. Järkevimpiä käyttökohteita sille ovat esimerkiksi tiedon muuttaminen eri muotoon tai formaattiin. Sitä kannattaa myös käyttää silloin, kun tietolähteitä on useampia ja ne eroavat toisistaan. Esimerkiksi käytössä voisi olla tiedostojärjestelmä, ftp, http ja Webservice. Biztalkilla voidaan yhdistää erilaisia tekniikoita.

Biztalk Serveriä ei kannata käyttää esimerkiksi yksinkertaiseen tietokantakyselyyn. Tälläisia tarpeita varten voidaan käyttää muita tekniikoita. Tietokantakyselystä voi

tehdä .Net-Webservicen. Tälläisissä tapauksissa ei tarvita Biztalkin kaltaista laajaa järjestelmää. Lisäksi Biztalk Serveriä ei kannata käyttää pelkästään yksittäisen tiedoston käsittelyyn.

7 JOHTOPÄÄTÖKSET

Tavoitteena oli saada kaksi tietolähdettä integroitua yhdeksi kokonaisuudeksi pelkästään Biztalkin sisällä. Tähän tavoitteeseen ei päästy suoraan. Ensimmäiseksi ongelmaksi muodostui Biztalk Server 2006 -tuotteen tarjoama SQL-adapteri. Tietokannasta tuntui olevan mahdotonta saada mitään tuloksia ulos. Ongelma korjaantui myöhemmin, kun käyttöön saatiin täysin uudistettu WCFcustom-adapteri. Toiseksi ongelmaksi muodostui AssetWebservicen kykenemättömyys vastata yleisimpien Webservice –palveluiden vaatimuksiin. Palvelulta puuttui kaksi normaalin Webservicen tunnusmerkkiä: Wsdl-kuvaus ja protokolla. Biztalk ei pysty käsittelemään sellaista palvelua, joka ei toteuta minkäänlaisia sovittuja yhteisiä sääntöjä.

AssetWebservicen kohdalla muutettiin käsittelytapa 3stepIT:n suosittelemaan tapaan. 3stepIT suosittelee, että haetaan kaikki tiedot omista laitteista palvelun kautta ja tallennetaan omaan tietokantaan. 3stepIt painotti, että päivitystä ei saisi tehdä virka-aikaan, koska muuten tietoliikenneyhteydet saattaisivat ruuhkautua.

Seuraavaksi aloitettiin ohjelman suunnittelu, joka toteuttaisi päivityksen yöaikaan C#-ohjelmointikielellä. Ohjelman kehittämisessä oli ensin jonkin verran ongelmia, koska AssetWebservicen SSL-sertifikaatti ei ollut validi. Ohjelmassa piti onnistua kiertämään sertifikaatti. Sertifikaatin kiertäminen oli kuitenkin lopuksi suhteellisen helppoa. Ohjelmaa ei vieläkään voi käyttää suoraan esimerkiksi Biztalkissa, koska palvelupyyntö kestää oman aikansa ennen kuin se menee läpi. AssetWebservicelle luotiin oma tietokanta, johon Biztalk voi olla yhteydessä.

Projektissa päästiin niihin tavoitteisiin, mitkä sille asetettiin aluksi. Projektin edetessä jouduttiin kehittämään erilaisia kiertotapoja, mutta se kuuluu ohjelmistokehitykseen.

LÄHTEET

Hunter, D., Watt, A., Rafter, J., Duckett, J., Ayers, D., Chase, N., Fawcet, J., Gaven & T., Patterson, B. 2004. Beginning XML 3rd edition. Indiapolis, Indiana: Wiley Publishing

Harold, ER. 2000. Tehokäyttäjän opas XML. Suomentaja Pekka Saxberg. Jyväskylä: Gummerus.

Vlist, E. 2002. XML Schema. California:O'Reilly.

Jefford, D., Smith, K. & Fairweather, E. 2007. Professional BizTalk Server 2006. Indiapolis, Indiana: Wiley Publishing.

Dunphy, G., Metwally, A. 2006. Pro Biztalk 2006. Berkeley, California: Apress.

Woolston, D. 2007. Foundations of Biztalk Server 2006. Carlifornia: Apress.

Ilenius, T. 2009. Järjestelmänhallinta System Center Configuration Managerilla.[Verkkojulkaisu]. Turku: Turun ammattikorkeakoulu. Opinnäytetyö. [Viitattu 21.8.2009]. Saatavana: http://www.escfi.net/KnowledgeBank/Public.aspx?kboId=79

System Center Configuration Manager: Overview. [WWW-dokumentti]. [Viitattu 12.9.2009]. Saatavissa:http://www.microsoft.com/systemcenter/configurationmanager/en/us/overview.aspx

How to Use the BizTalk Web Services Publishing Wizard to Publish an Orchestration as a Web Service. [WWW-dokumentti]. [Viitattu 13.9.2009]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa578703%28BTS.10%29.aspx

LIITTEET

Liite 1: ReadXml-metodi

```
public static string readXml(string filepath)
{
    //Alustetaan muuttuja rivejä varten
    string line;

    //Luodaan Stringbuilder objekti
    StringBuilder xmltied = new StringBuilder();

    //Tarkistetaan onko tiedosto olemassa
    if (File.Exists(filepath))
    {
        //Luodaan Streamreader
        StreamReader file = null;
        //Asetetaan try-finally lohko
        try
        {
            //Asetetaan Streamreaderiin tiedostonpolku
            file = new StreamReader(filepath);
            //Luetaan tiedostoa rivi kerrallaan
            //Tarkistetaan myös, että rivi ei ole tyhjä
            while ((line = file.ReadLine()) != null)
            {
                //Lisätään rivi Stringbuilder objektiin
                xmltied.Append(line);
                //Lisätään rivinvaihto Stringbuilder objektiin
                xmltied.Append("\n");
            }
        }
        //..ja lopuksi
        finally
        {
            //Jos tiedosto ei ole tyhjä niin suljetaan se
            if (file != null)
                file.Close();
        }
    }
    //Palautetaan Xml-tiedosto String-merkkijonona
    return xmltied.ToString();


}
```

Liite 2: MD5Encrypt-metodi

```
public static string MD5Encrypt(string valueString)
{
    //Alustetaan String-muuttuja, joka palautetaan
    string back = String.Empty;

    //Luodaan MD5CryptoServiceProvider-objekti
    MD5CryptoServiceProvider md5Hasher = new
   MD5CryptoServiceProvider();

   //Kovertoidaan parametrina tuleva String-muuttuja Byte-
   muuttujaksi
    //ASCII-merkistön mukaisesti
   byte[] data =
   System.Text.Encoding.ASCII.GetBytes(valueString);

    //Lasketaan MD5-hash Byte-muuttujasta
    data = md5Hasher.ComputeHash(data);

    //Konvertoidaan avain Base64-mukaiseksi String-jonoksi
    back = Convert.ToBase64String(data);
    //URL-enkoodataan avain
    back = HttpUtility.UrlEncode(back);
    //Palautetaan avain kutsujalle
    return back
}
```

Liite 3 (2): PostAndGetResponse-metodi

```
public static string PostAndGetResponse(string url, string xmlstring)

{
            //Luodaan URL-osoitteesta HttpWebRequest-objekti
            //WebRequest pitää pakottaa HttpWebRequest-objektiksi
            HttpWebRequest pyynto =
           (HttpWebRequest)WebRequest.Create(url);
            //Annetaan pyynnölle otsaketiedot
            pyynto.ContentType = "application/x-www-form-urlencoded";
            //Asetetaan metodiksi: POST
            pyynto.Method = "POST";

            //Ohitetaan AssetWebServicen sertifikaattitarkistus

           ServicePointManager.ServerCertificateValidationCallback =
           TrustAllCertificatesCallback;

            //Sijoitetaan Xml-merkkijono Byte-muuttujaan
            byte[] bytes = Encoding.ASCII.GetBytes(xmlstring);
            //Luodaan uusi Stream-muuttuja ja alustetaan se
            Stream os = null;

            //Sijoitetaan pyyntö try-catch-finally lohkoihin
            try
            {
                //Lasketaan pyynnön pituus
                pyynto.ContentLength = bytes.Length;
                //Tehdään pyyntö AssetWebServicelle
                os = pyynto.GetRequestStream();
                //Kirjoitetaan virtaan Xml-merkkijono ja virranpituus
                os.Write(bytes, 0, bytes.Length);
            }
            //Tarkistetaan mahdollinen Web-virhe
            catch (WebException ex)
            {
                Console.WriteLine(ex.Message, "HttpPost: Request error");
            }
            //..ja lopuksi
            finally
            {
                //Jos virta ei ole tyhjä
                if (os != null)
                {
                    //..suljetaan se
                    os.Close();
                }
            }

            //Vastauksen vastaanottaminen
            //Sijoitetaan try-catch lohkoon
            try
            {
                //Haetaan mahdollinen vastaus HttpWebResponse-objektin
                avulla
                HttpWebResponse vastaus =
                (HttpWebResponse)pyynto.GetResponse();
                //Jos vastaus on tyhjä, niin..
                if (vastaus == null)
```

```
    {   //..palautetaan tyhjä vastaus
        return null;
    }
    //Luodaan StreamReader-objekti Response-virrasta.
          StreamReader sr = new
          StreamReader(vastaus.GetResponseStream());
    //Palautetaan virta kutsujalle ja poistetaan tyhjät välit
    return sr.ReadToEnd().Trim();
}
    //Tarkistetaan mahdollinen Web-virhe
catch (WebException ex)
{
    //Annetaan asianmukainen virheilmoitus
Console.WriteLine(ex.Message, "HttpPost: Response error");
}
//Palautetaan tyhjä vastaus
return null;}
```

Liite 4 (5): ParseXmlAndDoDBupdate-metodi

```
public static void ParseXmlAndDoDBupdate(string xmlresponse)
{

            //Luodaan uusi XmlReaderSettings objekti
             XmlReaderSettings setting = new XmlReaderSettings();
             //Kirjataan asetuksiin, että DTD ei ole kielletty
             setting.ProhibitDtd = false;
             //Luodaan uusi XmlDocument-objekti
             XmlDocument doc = new XmlDocument();
             //Ladataan objektiin vastaanotettu Xml-virta, luetaan se
            XmlReader-objektiin
             //StringReader-objektin avulla, Lisätään myös tarvittavat
            asetukset
             doc.Load(XmlReader.Create(new StringReader(xmlresponse),
            setting));

             //Luodaan XmlNodeList, johon valitaan ne elementit, joiden
            alta halutaan tietoa
             XmlNodeList innerList = doc.SelectNodes("//export/entity");



            //Asetetaan yhteys-String tietokantaa varten
            string conStr = "Integrated Security=SSPI;Persist Security
            Info=False;Initial Catalog=AssetData;Data Source=BIZTALK";

             //Luodaan uusi SqlConnection yhteys-Stringin avulla
             using (SqlConnection con = new SqlConnection(conStr))
             {
                 //Luodaan uusi Sql-komento
                 SqlCommand command = con.CreateCommand();
                 //Luodaan komento, joka tyhjää Asset-tietokannan taulun
                 command.CommandText = "Truncate Table AssetData";
                 //Sijoitetaan suoritus try-catch lohkoon
                 try
                 {
                     //Avataan yhteys
                     con.Open();
                     //Suoritetaan komento, joka ei palauta mitään
                     command.ExecuteNonQuery();
                     //Suljetaan yhteys
                     con.Close();
                 }
                 //Tarkistetaan mahdollinen Sql-virhe
                 catch (SqlException sek)
                 {
                     //Annetaan asianmukaiset virheilmoitukset
                     Console.WriteLine("SqlError: " + sek.Message);
                 }
                 //Ilmoitetaan, että taulu on nyt tyhjä
                 Console.WriteLine("Table erase done!");

             }
             //Luodaan uusi Sql-yhteys
             using (SqlConnection con = new SqlConnection(conStr))
             {
                 //Käydään foreach-luupissa läpi XmlNode-lista
```

```
                    foreach (XmlNode oneObject in innerList)
                    {
                        //Luodaan uusi Sql-komento
                        SqlCommand command = con.CreateCommand();

                        //Luodaan Insert-lause kaikille laitteen tiedoille,
                        //sekä annetaan parametreina tiedot

                        command.CommandText = "Insert into AssetData"
+"(name,idnumber,serialnumber,integrationid,username,productgroupname,pro
cessor,"
+"memory,displaysize,storage,extratext1,extratext2,extranumber1,extranumb
er2,"
+"costcenter,location,version,extrainfo,project,companyproductgroup,extra
info1,"
+"extrainfo2,deviceprice,devicerent,contractnumber,length,startdate,endda
te,deliverydate,"
+"signingdate,interval,contractprice,terminated,endingoption,publicstatus
,changedate)"
+"Values(@name,@idnumber,@serialnumber,@integrationid,@username,@productg
roupname,@processor,"
+"@memory,@displaysize,@storage,@extratext1,@extratext2,@extranumber1,@ex
tranumber2,@costcenter,"+
"@location,@version,@extrainfo,@project,@companyproductgroup,@extrainfo1,
@extrainfo2,"
+"@deviceprice,@devicerent,@contractnumber,@length,@startdate,@enddate,@d
eliverydate,@signingdate,"
+"@interval,@contractprice,@terminated,@endingoption,@publicstatus,@chang
edate)";

//Lisätään parametreihin tiedot Xml-virrasta yksi kerrallaan Xpathin
avulla

command.Parameters.AddWithValue("@name",
oneObject.SelectSingleNode("attribute[@code='name']").InnerText);
command.Parameters.AddWithValue("@idnumber",
oneObject.SelectSingleNode("attribute[@code='deviceid']").InnerText);
command.Parameters.AddWithValue("@serialnumber",
oneObject.SelectSingleNode("attribute[@code='serialnumber']").InnerText.T
rim());
command.Parameters.AddWithValue("@integrationid",
oneObject.SelectSingleNode("attribute[@code='integrationid']").InnerText)
;
command.Parameters.AddWithValue("@username",
oneObject.SelectSingleNode("attribute[@code='username']").InnerText);
command.Parameters.AddWithValue("@productgroupname",
oneObject.SelectSingleNode("attribute[@code='productgroupname']").InnerTe
xt);
command.Parameters.AddWithValue("@processor",
oneObject.SelectSingleNode("attribute[@code='processor']").InnerText);
 command.Parameters.AddWithValue("@memory",
oneObject.SelectSingleNode("attribute[@code='memory']").InnerText);
command.Parameters.AddWithValue("@displaysize",
oneObject.SelectSingleNode("attribute[@code='displaysize']").InnerText);
command.Parameters.AddWithValue("@storage",
oneObject.SelectSingleNode("attribute[@code='storage']").InnerText);
command.Parameters.AddWithValue("@extratext1",
oneObject.SelectSingleNode("attribute[@code='extratext1']").InnerText);
command.Parameters.AddWithValue("@extratext2",
oneObject.SelectSingleNode("attribute[@code='extratext2']").InnerText);
```

```
command.Parameters.AddWithValue("@extranumber1",
oneObject.SelectSingleNode("attribute[@code='extranumber1']").InnerText);
command.Parameters.AddWithValue("@extranumber2",
oneObject.SelectSingleNode("attribute[@code='extranumber2']").InnerText);
command.Parameters.AddWithValue("@costcenter",
oneObject.SelectSingleNode("attribute[@code='costcenter']").InnerText);
command.Parameters.AddWithValue("@location",
oneObject.SelectSingleNode("attribute[@code='location']").InnerText);
command.Parameters.AddWithValue("@version",
oneObject.SelectSingleNode("attribute[@code='version']").InnerText);
command.Parameters.AddWithValue("@extrainfo",
oneObject.SelectSingleNode("attribute[@code='extrainfo']").InnerText);
command.Parameters.AddWithValue("@project",
oneObject.SelectSingleNode("attribute[@code='project']").InnerText);
command.Parameters.AddWithValue("@companyproductgroup",
oneObject.SelectSingleNode("attribute[@code='companyproductgroup']").Inne
rText);
command.Parameters.AddWithValue("@extrainfo1",
oneObject.SelectSingleNode("attribute[@code='extrainfo1']").InnerText);
command.Parameters.AddWithValue("@extrainfo2",
oneObject.SelectSingleNode("attribute[@code='extrainfo2']").InnerText);
command.Parameters.AddWithValue("@deviceprice",
oneObject.SelectSingleNode("attribute[@code='deviceprice']").InnerText);
command.Parameters.AddWithValue("@devicerent",
oneObject.SelectSingleNode("attribute[@code='devicerent']").InnerText);
command.Parameters.AddWithValue("@contractnumber",
oneObject.SelectSingleNode("attribute[@code='contractnumber']").InnerText
);
command.Parameters.AddWithValue("@length",
oneObject.SelectSingleNode("attribute[@code='length']").InnerText);
command.Parameters.AddWithValue("@startdate",
oneObject.SelectSingleNode("attribute[@code='startdate']").InnerText);
command.Parameters.AddWithValue("@enddate",
oneObject.SelectSingleNode("attribute[@code='enddate']").InnerText);
command.Parameters.AddWithValue("@deliverydate",
oneObject.SelectSingleNode("attribute[@code='deliverydate']").InnerText);
command.Parameters.AddWithValue("@signingdate",
oneObject.SelectSingleNode("attribute[@code='signingdate']").InnerText);
command.Parameters.AddWithValue("@interval",
oneObject.SelectSingleNode("attribute[@code='interval']").InnerText);
command.Parameters.AddWithValue("@contractprice",
oneObject.SelectSingleNode("attribute[@code='contractprice']").InnerText)
;
command.Parameters.AddWithValue("@terminated",
oneObject.SelectSingleNode("attribute[@code='terminated']").InnerText);
command.Parameters.AddWithValue("@endingoption",
oneObject.SelectSingleNode("attribute[@code='endingoption']").InnerText);
command.Parameters.AddWithValue("@publicstatus",
oneObject.SelectSingleNode("attribute[@code='publicstatus']").InnerText);
command.Parameters.AddWithValue("@changedate",
oneObject.SelectSingleNode("attribute[@code='changedate']").InnerText);

                    //Sijoitetaan suoritus try-catch lohkoon
                    try
                    {
                        //Avataan yhteys
                        con.Open();
                        //Suoritetaan komento, joka ei palauta mitään
                        command.ExecuteNonQuery();
                        //Suljetaan yhteys
                        con.Close();
```

```
            }
            //Tarkistetaan mahdollinen Sql-virhe
            catch (SqlException sek)
            {
                //Annetaan asianmukainen virheilmoitus
                Console.WriteLine("SqlError: " + sek.Message);
            }
              //Kirjoitetaan konsoliin jokaisen lisätyn laitteen
              nimi

Console.WriteLine(oneObject.SelectSingleNode("attribute[@code=
'name']").InnerText + " added!");
        }
    }
    //Avataan uusi Sql-yhteys
    using (SqlConnection con = new SqlConnection(conStr))
    {
        //Luodaan uusi Sql-komento
        SqlCommand command = con.CreateCommand();
        //Luodaan Sql-komento, joka tyhjää Updated taulun
        command.CommandText = "Truncate Table Updated";
        //Sijoitetaan suoritus try-catch lohkoon
        try
        {
            //Avataan yhteys
            con.Open();
            //Suoritetaan komento, joka ei palauta mitään
            command.ExecuteNonQuery();
            //Suljetaan yhteys
            con.Close();
        }
        //Tarkistetaan mahdollinen Sql-virhe
        catch (SqlException sek)
        {
            //Annetaan asianmukainen virheilmoitus
            Console.WriteLine("SqlError: " + sek.Message);
        }
    }
    //Avataan uusi Sql-yhteys
    using (SqlConnection con = new SqlConnection(conStr))
    {
        //Haetaan sen hetkinen kellonaika ja pvm. muuttujaan
        string datetime = DateTime.Now.ToString();
        //Luodaan uusi Sql-komento
        SqlCommand command = con.CreateCommand();
        //Luodaan Sql-komento, joka lisää Updated-tauluun
        päivitysajan
        command.CommandText = "Insert Into Updated (LastUpdated)
        Values (@datetime)";
        //Lisätään aika komennon parametreihin
        command.Parameters.AddWithValue("@datetime", datetime);
        //Sijoitetaan suoritus try-catch lohkoon
        try
        {
            //Avataan yhteys
            con.Open();
            //Suoritetaan komento, joka ei palauta mitään
            command.ExecuteNonQuery();
            //Suljetaan yhteys
            con.Close();
        }
```

```
            //Tarkistetaan mahdollinen Sql-virhe
            catch (SqlException sek)
            {
                //Annetaan asianmukainen virheilmoitus
                Console.WriteLine("SqlError: " + sek.Message);
            }
        }


    }
```

Liite 5 (2): GetAssetDeviceTypedProcedure.dbo-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
xmlns:ns3="http://schemas.microsoft.com/Sql/2008/05/ProceduresResultSets/dbo/sp_Get
AssetDevice" elementFormDefault="qualified" targetNames-
pace="http://schemas.microsoft.com/Sql/2008/05/TypedProcedures/dbo" version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import schemaLoca-
tion=".\GetAssetDeviceProcedureResultSet.dbo.sp_GetAssetDevice.xsd" names-
pace="http://schemas.microsoft.com/Sql/2008/05/ProceduresResultSets/dbo/sp_GetAssetD
evice" />
  <xs:annotation>
    <xs:appinfo>
      <fileNameHint
xmlns="http://schemas.microsoft.com/servicemodel/adapters/metadata/xsd">TypedProced
ure.dbo</fileNameHint>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="sp_GetAssetDevice">
    <xs:annotation>
      <xs:documentation>
        <doc:action
xmlns:doc="http://schemas.microsoft.com/servicemodel/adapters/metadata/documentation
">TypedProcedure/dbo/sp_GetAssetDevice</doc:action>
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1" name="serialnumber" nillable="true">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="50" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="sp_GetAssetDeviceResponse">
    <xs:annotation>
      <xs:documentation>
        <doc:action
xmlns:doc="http://schemas.microsoft.com/servicemodel/adapters/metadata/documentation
">TypedProcedure/dbo/sp_GetAssetDevice/response</doc:action>
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
```

```
      <xs:element minOccurs="0" maxOccurs="1" name="StoredProcedureResultSet0"
nillable="true" type="ns3:ArrayOfStoredProcedureResultSet0" />
      <xs:element minOccurs="1" maxOccurs="1" name="ReturnValue" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Liite 6 (3): GetAssetDeviceProcedureResultSet.dbo-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
xmlns:ns3="http://schemas.microsoft.com/Sql/2008/05/ProceduresResultSets/dbo/sp_Get
AssetDevice" elementFormDefault="qualified" targetNames-
pace="http://schemas.microsoft.com/Sql/2008/05/ProceduresResultSets/dbo/sp_GetAssetD
evice" version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
      <fileNameHint
xmlns="http://schemas.microsoft.com/servicemodel/adapters/metadata/xsd">ProcedureRes
ultSet.dbo.sp_GetAssetDevice</fileNameHint>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="StoredProcedureResultSet0">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="asset_id" nillable="true"
type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="name" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="idnumber" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="serialnumber" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="integrationid" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="username" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="productgroupname" nilla-
ble="true" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="processor" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="memory" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="displaysize" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="storage" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="extratext1" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="extratext2" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="extranumber1" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="extranumber2" nillable="true"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="costcenter" nillable="true"
type="xs:string" />
```

```
    <xs:element minOccurs="0" maxOccurs="1" name="location" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="version" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="extrainfo" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="project" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="companyproductgroup" nilla-
ble="true" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="extrainfo1" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="extrainfo2" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="deviceprice" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="devicerent" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="contractnumber" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="length" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="startdate" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="enddate" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="deliverydate" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="signingdate" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="interval" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="contractprice" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="terminated" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="endingoption" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="publicstatus" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="changedate" nillable="true"
type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="LastUpdated" nillable="true"
type="xs:string" />
   </xs:sequence>
  </xs:complexType>
  <xs:element name="StoredProcedureResultSet0" nillable="true"
type="ns3:StoredProcedureResultSet0" />
  <xs:complexType name="ArrayOfStoredProcedureResultSet0">
   <xs:sequence>
```

```
      <xs:element minOccurs="0" maxOccurs="unbounded"
name="StoredProcedureResultSet0" type="ns3:StoredProcedureResultSet0" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfStoredProcedureResultSet0" nillable="true"
type="ns3:ArrayOfStoredProcedureResultSet0" />
</xs:schema>
```

Liite 7 (2): WCF Send Port Custom Binding Xml

```
<?xml version="1.0" encoding="utf-8"?>
<BindingInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Assem-
bly="Microsoft.BizTalk.Deployment, Version=3.0.1.0, Culture=neutral, PublicKeyTo-
ken=31bf3856ad364e35" Version="3.5.1.0" BindingStatus="NoBindings" BoundEnd-
points="0" TotalEndpoints="0">
  <Timestamp>2009-07-02T17:55:28.017392+03:00</Timestamp>
  <ModuleRefCollection />
  <SendPortCollection>
    <SendPort Name="AssetSqlSend" IsStatic="true" IsTwoWay="true" BindingOp-
tion="0">
      <Description>SendPort for SqlAdapterBinding.</Description>
      <TransmitPipeline Name="Microsoft.BizTalk.DefaultPipelines.XMLTransmit" Fully-
QualifiedName="Microsoft.BizTalk.DefaultPipelines.XMLTransmit, Micro-
soft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyTo-
ken=31bf3856ad364e35" Type="2" TrackingOption="None" Description="" />
      <PrimaryTransport>
        <Address>mssql://biztalk//AssetData?</Address>
        <TransportType Name="WCF-Custom" Capabilities="907" ConfigurationCls-
id="af081f69-38ca-4d5b-87df-f0344b12557a" />
        <TransportTypeData>&lt;CustomProps&gt;
  &lt;BindingType vt="8"&gt;sqlBinding&lt;/BindingType&gt;
  &lt;BindingConfiguration vt="8"&gt;&amp;lt;binding name="SqlAdapterBinding" clo-
seTimeout="00:01:00" openTimeout="00:01:00" receiveTimeout="00:10:00" sendTi-
meout="00:01:00" maxConnectionPoolSize="100" encrypt="false" workstationId=""
useAmbientTransaction="true" batchSize="20" polledDataAvailableStatement="" pol-
lingStatement="" pollingIntervalInSeconds="30" pollWhileDataFound="false" notifica-
tionStatement="" notifyOnListenerStart="true" enableBizTalkCompatibilityMode="true"
chunkSize="4194304" inboundOperationType="Polling" useDatabaseNameInXsdNames-
pace="false" allowIdentityInsert="false" enablePerformanceCounters="false" xmlStored-
ProcedureRootNodeName="" xmlStoredProcedureRootNodeNamespace=""
/&amp;gt;&lt;/BindingConfiguration&gt;
  &lt;StaticAction vt="8"&gt;&amp;lt;BtsActionMapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"&amp;gt;
  &amp;lt;Operation Name="sp_GetAssetDevice" Ac-
tion="TypedProcedure/dbo/sp_GetAssetDevice" /&amp;gt;
&amp;lt;/BtsActionMapping&amp;gt;&lt;/StaticAction&gt;
  &lt;UseSSO vt="11"&gt;0&lt;/UseSSO&gt;
  &lt;InboundBodyLocation vt="8"&gt;UseBodyElement&lt;/InboundBodyLocation&gt;
  &lt;InboundNodeEncoding vt="8"&gt;Xml&lt;/InboundNodeEncoding&gt;
  &lt;OutboundBodyLocation
vt="8"&gt;UseBodyElement&lt;/OutboundBodyLocation&gt;
  &lt;OutboundXmlTemplate vt="8"&gt;&amp;lt;bts-msg-body
xmlns="http://www.microsoft.com/schemas/bts2007" encod-
ing="xml"/&amp;gt;&lt;/OutboundXmlTemplate&gt;
```

```
  &lt;PropagateFaultMessage vt="11"&gt;-1&lt;/PropagateFaultMessage&gt;
  &lt;Identity vt="8" /&gt;
&lt;/CustomProps&gt;</TransportTypeData>
        <RetryCount>3</RetryCount>
        <RetryInterval>5</RetryInterval>
        <ServiceWindowEnabled>false</ServiceWindowEnabled>
        <FromTime>2000-01-01T00:00:00</FromTime>
        <ToTime>2000-01-01T23:59:59</ToTime>
        <Primary>true</Primary>
        <OrderedDelivery>false</OrderedDelivery>
        <DeliveryNotification>0</DeliveryNotification>
        <SendHandler xsi:nil="true" />
      </PrimaryTransport>
      <SecondaryTransport>
        <Address />
        <TransportTypeData />
        <RetryCount>3</RetryCount>
        <RetryInterval>5</RetryInterval>
        <ServiceWindowEnabled>false</ServiceWindowEnabled>
        <FromTime>2000-01-01T00:00:00</FromTime>
        <ToTime>2000-01-01T23:59:59</ToTime>
        <Primary>false</Primary>
        <OrderedDelivery>false</OrderedDelivery>
        <DeliveryNotification>0</DeliveryNotification>
        <SendHandler xsi:nil="true" />
      </SecondaryTransport>
      <ReceivePipeline Name="Microsoft.BizTalk.DefaultPipelines.XMLReceive" Fully-
QualifiedName="Microsoft.BizTalk.DefaultPipelines.XMLReceive, Micro-
soft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyTo-
ken=31bf3856ad364e35" Type="1" TrackingOption="None" Description="" />
      <ReceivePipelineData xsi:nil="true" />
      <Tracking>0</Tracking>
      <Filter />
      <OrderedDelivery>false</OrderedDelivery>
      <Priority>5</Priority>
      <StopSendingOnFailure>false</StopSendingOnFailure>
      <RouteFailedMessage>false</RouteFailedMessage>
      <ApplicationName xsi:nil="true" />
    </SendPort>
  </SendPortCollection>
  <DistributionListCollection />
  <ReceivePortCollection />
  <PartyCollection />
</BindingInfo>
```

Liite 8 (3): Webservicen käyttöliittymä Asp.Net ympäristössä

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="WsClient.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Asset-Sms Integraatio</title>
<link href="tyyli.css" title="compact" rel="stylesheet" type="text/css"/>
</head>
<body>
    <form id="form1" runat="server">
    <div class="content">
      <center>
      <b class="otsikko">Asset-Sms Integraatio</b><br /><br />
           <i class="selite">Haku sarjanumerolla:
           esim."YKBX036392","YK8D082132"</i><br /><br />
             
          <asp:TextBox ID="Hakuparametri" runat="server"></asp:TextBox>
           
           <asp:Button ID="hae" runat="server" Text="Hae"
           onclick="hae_Click"/><br />
        <asp:Label ID="virhe" runat="server"
        CssClass="virhe"></asp:Label>
      </center>
    </div><br />
    <center>
<asp:Table runat="server" ID="resultset1" style="display:none"
Width="500">
       <asp:TableHeaderRow CssClass="totsikko"><asp:TableHeaderCell
ColumnSpan="2">SMS-tietokanta</asp:TableHeaderCell></asp:TableHeaderRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Valmistaja:</asp:TableCell><asp:TableCell><
asp:Label runat="server"
ID="valmistaja"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Kuvaus:</asp:TableCell><asp:TableCell><asp
:Label runat="server"
ID="kuvaus"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Julkaisupäivä:</asp:TableCell><asp:TableCel
l><asp:Label runat="server"
ID="julkaisu"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Lähdeid:</asp:TableCell><asp:TableCell><as
p:Label runat="server"
ID="smsid"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Sarjanumero:</asp:TableCell><asp:TableCell>
<asp:Label runat="server"
ID="sarjanro"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="toka"><asp:TableCell>SMS Bios
versio:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="biosversion"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Ohjelmisto:</asp:TableCell><asp:TableCell><
```

```
asp:Label runat="server"
ID="ohjelmisto"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Aikaleima:</asp:TableCell><asp:TableCell><
asp:Label runat="server"
ID="aika"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="eka"><asp:TableCell>SMS
versio:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="smsversio"></asp:Label></asp:TableCell></asp:TableRow>
       </asp:Table><br />
       <asp:Table runat="server" ID="resultset2" style="display:none"
Width="500">
       <asp:TableHeaderRow CssClass="totsikko"><asp:TableHeaderCell
ColumnSpan="2">Asset-
tietokanta</asp:TableHeaderCell></asp:TableHeaderRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Muutospäivä:</asp:TableCell><asp:TableCell>
<asp:Label runat="server"
ID="muutos"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Sopimusnumero:</asp:TableCell><asp:TableCe
ll><asp:Label runat="server"
ID="sopimus"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Sopimushinta:</asp:TableCell><asp:TableCell
><asp:Label runat="server"
ID="shinta"></asp:Label>€</asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Kustannuspaikka:</asp:TableCell><asp:Table
Cell><asp:Label runat="server"
ID="kustannus"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="eka"><asp:TableCell>Laitteen
hinta:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="laitehinta"></asp:Label>€</asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="toka"><asp:TableCell>Laitteen
vuokra:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="laitevuokra"></asp:Label>€</asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="eka"><asp:TableCell>Näytön
koko:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="naytto"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="toka"><asp:TableCell>Loppu
pvm.:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="loppupvm"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="eka"><asp:TableCell>Asset
Id:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="assetid"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Pituus:</asp:TableCell><asp:TableCell><asp
:Label runat="server"
ID="pituus"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Muisti:</asp:TableCell><asp:TableCell><asp:
Label runat="server"
ID="muisti"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Nimi:</asp:TableCell><asp:TableCell><asp:L
abel runat="server" ID="nimi"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Prosessori:</asp:TableCell><asp:TableCell><
asp:Label runat="server"
ID="prosessori"></asp:Label></asp:TableCell></asp:TableRow>
```

```
        <asp:TableRow
CssClass="toka"><asp:TableCell>Laiteryhmä:</asp:TableCell><asp:TableCell>
<asp:Label runat="server"
ID="laiteryh"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Sarjanumero:</asp:TableCell><asp:TableCell>
<asp:Label runat="server"
ID="asarjanro"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Allekirjoituspäivä:</asp:TableCell><asp:Ta
bleCell><asp:Label runat="server"
ID="allekirjoitus"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Alkupäivä:</asp:TableCell><asp:TableCell><a
sp:Label runat="server"
ID="alkupaiva"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="toka"><asp:TableCell>Varasto:</asp:TableCell><asp:TableCell><as
p:Label runat="server"
ID="varasto"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow
CssClass="eka"><asp:TableCell>Käyttäjänimi:</asp:TableCell><asp:TableCell
><asp:Label runat="server"
ID="kayttaja"></asp:Label></asp:TableCell></asp:TableRow>
        <asp:TableRow CssClass="toka"><asp:TableCell>Asset tietokanta
päivitetty:</asp:TableCell><asp:TableCell><asp:Label runat="server"
ID="paivitys"></asp:Label></asp:TableCell></asp:TableRow>
       </asp:Table>

    </center>
    </form>
    </body>
</html>
```

Liite 9: Lopullinen asiakasohjelma

Asset-Sms Integraatio

Haku sarjanumerolla: esim."YKBX036392","YK8D082132"

YKBX036392 Hae

SMS-tietokanta	
Valmistaja:	Phoenix
Kuvaus:	Ver 1.00PARTTBL
Julkaisupäivä:	2007-10-26T00:00:00Z
Lähdeid:	43370
Sarjanumero:	YKBX036392
SMS Bios versio:	1.03 - 041 - 1471
Ohjelmisto:	Ver 1.00PARTTBL
Aikaleima:	2008-09-12T11:23:15Z
SMS versio:	FSC - 6040000
Asset-tietokanta	
Muutospäivä:	2009-03-03
Sopimusnumero:	007256-AD03-001-000
Sopimushinta:	2065 €
Kustannuspaikka:	20 Yhteiset palvelut, Tietohallinto
Laitteen hinta:	1030 €
Laitteen vuokra:	86.36 €
Näytön koko:	0
Loppu pvm.:	2011-06-30
Asset Id:	160238140
Pituus:	36
Muisti:	0
Nimi:	ESPRIMO Mobile D9500/3Y Onsite
Prosessori:	0
Laiteryhmä:	Laptops
Sarjanumero:	YKBX036392
Allekirjoituspäivä:	2008-04-18
Alkupäivä:	2008-07-01
Varasto:	0
Käyttäjänimi:	Alanko Arja (luottamusmies)
Asset tietokanta päivitetty:	2.9.2009 0:01:45