

Sami Kettunen

TOIMINNANOHJAUSJÄRJESTEL- MIEN INTEGRAATIO

Opinnäytetyö
Tietojenkäsittely


Toukokuu 2012




MIKKELIN AMMATTIKORKEAKOULU

Mikkeli University of Applied Sciences

KUVAILULEHTI

 <p>MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences</p>		Opinnäytetyön päivämäärä 14.5.2012	
Tekijä(t) Sami Kettunen		Koulutusohjelma ja suuntautuminen Tietojenkäsittely	
Nimeke Toiminnanohjausjärjestelmien integraatio			
Tiivistelmä <p>Tarkoituksenani tässä työssä oli tutustua taloushallinnon digitalisoitumiseen ja prosesseihin, toiminnanohjausjärjestelmiin yleisellä tasolla ja näiden järjestelmien välisten integraatioiden mahdollistaviin teknologioihin. Työskentely Epicor 9:n ja MediusFlow:n välisen integraation parissa ohjasi tätä tutkimusta, sillä halusin oma-aloitteisesti tutustua digitaaliseen taloushallintoon ja siihen liittyviin prosesseihin. Perehtyminen näihin prosesseihin auttoi minua integraation toteutuksessa. Toinen osa-alue oli tutkia järjestelmäintegraatioiden perusidea, mikä auttoi minua ymmärtämään paremmin kokonaisuutta.</p> <p>Tutkimus osoitti, että taloushallinnon digitalisoitumisen myötä tarve järjestelmien välisille integraatioille tulee säilymään myös tulevaisuudessa, sillä erilaisia järjestelmiä on niin paljon. Jotta järjestelmistä saatava hyöty olisi mahdollisimman suuri, tulee niiden kyetä yhteistyöhön muiden järjestelmien kanssa niin yrityksen sisällä kuin yrityksen välillä.</p> <p>Tutkimus osoitti myös, että integraatioteknologiat kehittyvät vauhdilla. Standardien avulla pyritään helpottamaan järjestelmäintegraatioiden toteutusta. Pitkälle kehittyneet integraatiosovellukset tarjoavat alustan järjestelmäintegraatioille. Tässä opinnäytetyössä toteutettu järjestelmäintegraatio on toteutettu Epicorin ja Mediuksen toteuttamien integraatiosovellusten avulla.</p>			
Asiasanat (avainsanat) XML, integraatio, integrointi, toiminnanohjaus, digitalisoituminen, sähköinen taloushallinto			
Sivumäärä 67 s. + liitteet 14 s.	Kieli Suomi	URN	
Huomautus (huomautukset liitteistä)			
Ohjaavan opettajan nimi Jukka Selin		Opinnäytetyön toimeksiantaja Smart Time Oy	

DESCRIPTION

 <p>MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences</p>		Date of the bachelor's thesis 14.5.2012	
Author(s) Sami Kettunen		Degree programme and option Business Information Technology	
Name of the bachelor's thesis Integration between ERP-systems			
Abstract <p>The goal of this bachelor's thesis was to find out the basics of processes in the digitalization of financial management. Also, my goal was to find out the basics of ERP-systems and technologies facilitating the integration between these systems. The study included work with the integration project between Epicor 9 and MediusFlow which revealed more about the processes of financial management. This was important for the successful integration between these two systems. A further part of my study was to find out the basics of system integrations which helped me to understand them as a whole.</p> <p>The study indicated that due to the digitalization of financial management and the variety of different systems the need for system integrations would stay in the future as well. To maximize the benefits that business systems provide it is crucial for them to be able to work together and transfer information within and between businesses.</p> <p>The study also indicated that integration technologies would be developing fast. Standards provide a way to make the integration between different systems easier to accomplish. Sophisticated integration applications have been developed over the years to facilitate system integrations. Integration between systems was accomplished with integration applications developed by Epicor and Medius.</p>			
Subject headings, (keywords) XML, integration, Enterprise Resource Planning, ERP, digitalization, e-commerce			
Pages 67 pages + appendices 14 pages		Language Finnish	
URN			
Remarks, notes on appendices			
Tutor Jukka Selin		Bachelor's thesis assigned by Smart Time Oy	

SISÄLTÖ

1	JOHDANTO	1
2	TALOUSHALLINNON DIGITALISOITUMINEN	2
2.1	Digitaalisen taloushallinnon hyödyt	3
2.2	Taloushallintojärjestelmät	4
2.3	Taloushallinnon prosessit	5
2.4	Sähköinen lasku ja ostolaskuprosessi	6
2.5	Kulu- ja matkalaskuprosessi	9
3	TOIMINNANOHJAUSJÄRJESTELMÄT	10
4	JÄRJESTELMÄINTEGRAATIOT	14
4.1	Haasteet	15
4.2	Palvelukeskeinen arkkitehtuuri (SOA)	17
4.3	Informaatiokeskeinen integraatio	19
4.4	liiketoimintaprosessien integraatio	20
4.5	Web Servicet	21
5	XML	22
5.1	SGML	23
5.2	XML:n vahvuudet	24
5.3	XML ja sen heikkoudet	25
5.4	XPath	26
5.5	XML Schema	27
5.6	XSLT	28
5.7	Simple Object Access Protocol (SOAP)	29
5.8	Web Service Description Language (WSDL)	30
6	INTEGRAATIOSOVELLUKSET	31
6.1	Epicor Service Connect	34
6.1.1	Workflow Designer	35
6.1.2	Komponentit Workflow Designerissa	36
6.1.3	Schema Utility	44
6.1.4	XML Mapper	47
6.2	Medius Integration Gateway	50
7	EPICOR 9–MEDIUSFLOW-INTEGRAATIO	52

7.1	Perustietojen ylläpito	53
7.2	Esimerkki: M1_E9_Medius_Supplier	56
7.2.1	Triggerin luominen	58
7.2.2	Output-kanavan luominen.....	61
7.2.3	Workflow-prosessi.....	63
7.2.4	MsgMedius_Supplier.xml.....	65
7.3	Muut rajapinnat.....	66
8	PÄÄTÄNTÖ	67
	LÄHTEET.....	68

1 JOHDANTO

Yritysten toimintojen hallinta on muuttumassa sähköisestä täysin digitaaliseen muotoon. Tämän kehityksen johdosta vuosien varrella kehitetyt järjestelmät ovat kovien paineiden alla, kun niiden tulee suoriutua toimialoittain muuttuvista erityisvaatimuksista. Myös yritysten taloushallinto on siirtymässä sähköisestä digitaaliseen muotoon. Taloushallinnon digitalisoinnin tavoitteena on kokonaan paperittomat prosessit aina laskutuksesta kirjanpitoon. Tässä kehityksessä vanha sanonta, ”mitä isot edeltä, sitä pienemmät perästä”, pitää hyvin paikkansa; järjestelmien hintojen pudotessa taloushallintoon digitalisoi yhä pienemmät yritykset. Täysin paperittomaan maailmaan on vielä erittäin pitkä matka, mutta Suomi on edelläkävijä myös laskuprosessien digitalisoimisessa.

Vaikka järjestelmät ovat nykyään jo parhaimmillaan hyvin joustavia ja vastaavat monipuolisesti yritysten vastaamiin erityistarpeisiin, ei niissäkään aina ole aivan kaikkia yrityksen prosesseja tukevia toimintoja. Tämä johtaa siihen, että yritykset joutuvat edelleen turvautumaan erilaisiin järjestelmäratkaisuihin täyttääkseen kaikki tarpeensa. Näin syntyy tarve näiden järjestelmien integroimiselle toisiinsa. Järjestelmäintegraatiohin on olemassa nykyään monenlaisia ratkaisuja, joista jokaisella on sekä hyvät että huonot puolensa. Järjestelmien integroiminen toisiinsa on hyvin haastavaa, sillä olemassa olevien järjestelmien määrä on valtava eivätkä ne koskaan ole täysin samanlaisia. Lisäksi järjestelmät on kehitetty vuosien varrella kovaa vauhtia kehittyvillä tekniikoilla, joten monet saattavat olla vanhoja jo syntyessään. Ennen vuosituhaten vaihetta yritykset päivittivät järjestelmiään kovaa vauhtia y2k:ta varten. Järjestelmien integroiminen toisiinsa ei ole suinkaan mahdotonta eikä kyseessä ole mikään uusi asia.

Tämän opinnäytetyön tutkimusongelmana oli suunnitella ja toteuttaa integraatio kahden eri toiminnanohjausjärjestelmän, MediusFlow:n ja Epicor 9:n, välille. Integraation onnistunut toteutus vaati tutustumista MediusFlow:hun, Medius Integration Gateway-integraatiosovellukseen ja taloushallintoon sekä laskutukseen liittyviin prosesseihin kuten hyväksyntä, laskujen tiliöinti ja kirjaaminen kirjanpitoon. Lisäksi tavoitteena oli toteuttaa integraatio siten, että sen voisi tarjota asiakkaalle mahdollisimman valmiina pakettina.

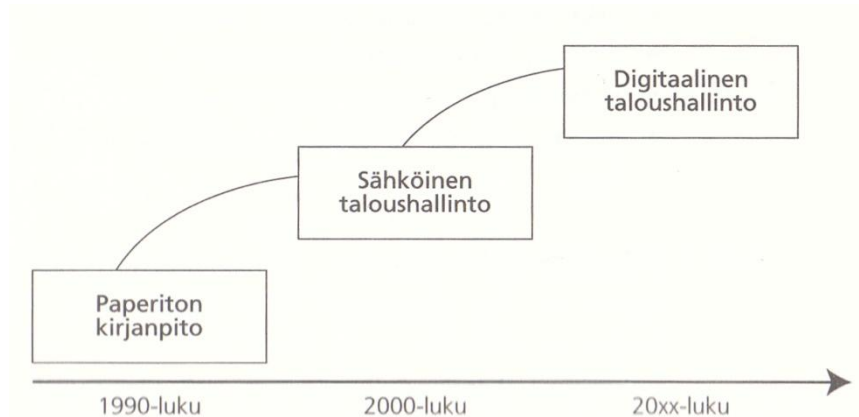
Luvussa kaksi kerron lyhyesti taloushallinnosta ja siitä, mitä sen digitalisoituminen tarkoittaa ja mikä sen rooli on suhteessa järjestelmäintegraatioihin. Luvussa kolme kuvailen lyhyesti mitä ovat toiminnanohjausjärjestelmät. Luvussa neljä kerrotaan tarkemmin mitä tarkoittaa järjestelmäintegraatio, minkälaisia tekniikoita järjestelmäintegraatioiden toteutukseen nykyään on olemassa, mihin suuntaan järjestelmäintegraatiot ovat kehittymässä ja mikä niiden tulevaisuus on. Koska XML-pohjaiset tekniikat ovat nykyään niin olennainen osa järjestelmäintegraatioita, kerron niistä kappaleessa viisi. Luvussa kuusi käsitelen tarkemmin integraatiosovelluksia, Epicor Service Connectia ja Medius Integration Gatewayta, koska niiden avulla tämän tässä opinnäytetyössä esiteltävä järjestelmäintegraatio on toteutettu. Seitsemännessä luvussa on kuvattu tämän opinnäytetyön osana toteutettu järjestelmäintegraatio.

2 TALOUSHALLINNON DIGITALISOITUMINEN

Sähköinen taloushallinto on jo tietotekniikan kehityksen huomioon ottaen vanha käsite, joka juontaa Suomessakin juurensa 1990-luvun alkupuoliskolle saakka. Tuolloin PC-tietokoneet yleistyivät ja taloushallintoon erikoistuneet ohjelmistot tulivat yhä pienempien yritysten ulottuville laitteistojen ja ohjelmistojen hintojen pudotessa. Lahden ja Salmisen mukaan (2008, 9) sähköisen taloushallinnon määritelmä riippuu siitä, kuka asian määrittelee tai missä yhteydessä siitä puhutaan. Useimmiten se nähdään suppeasti vain verkkolaskutuksena ja sähköisenä laskujen käsittelynä. Sana digitaalinen vie taloushallinnon yhä pidemmälle tietokoneiden maailmaan. Tavoitteena on pyrkiä kokonaan irti paperin käytöstä ja hyödyntää kovalla vauhdilla kehittyvien tietoverkkojen tarjoama nopeus. Taloushallinnon digitalisointi tuo tehokkuutta talouden hallintaan, laskujen käsittelyyn ja organisaatioiden prosesseihin.

Yrityksen toiminnassa keskeisistä osista on taloushallinto, jonka yksi osa on laskutus ja yritykselle esimerkiksi tavarantoimittajilta saapuvien laskujen käsittely. Kuten kaikessa yritystoiminnassa, myös taloushallinnossa tietotekniikan merkitys on korostunut Internetin yleistymisen myötä. Lähes poikkeuksetta ja yrityksen koosta riippumatta jokaisella yrityksellä tai organisaatiolla on tänä päivänä käytössään jonkinlainen sähköinen taloushallinto- tai kirjanpitojärjestelmä. (Lahti & Salminen 2008, 30)

Kuten kuvasta 2 voidaan tulkita, on taloushallinnon kehitys paperittomasta digitaaliseen tällä hetkellä sähköisen ja digitaalisen taloushallinnon murrosvaiheessa. Täysin digitaaliseen taloushallintoon on vielä pitkä matka. Rajoituksia täysin paperittomalle ja digitaaliselle taloushallinnolle asettavat vielä mm. järjestelmien puutteet ja niiden hinnat suhteessa niistä saatavaan hyötyyn.



KUVA 1. Taloushallinnon kehitys paperittomasta digitaaliseen (Lahti & Salminen 2008)

2.1 Digitaalisen taloushallinnon hyödyt

Salmisen ja Lahden mukaan (2008, 27) taloushallinnon digitaalisuus tuo talouden hallintaan huomattavia etuja. Digitaalisuus tarkoittaa sähköisessä muodossa olevan tiedon käsittelyä, siirtämistä, esittämistä ja varastointia. Sen ehdottomia etuja ovat sen tehokkuus ja nopeus. Samanaikaisesti resurssien ja arkistointitilan tarve vähenee huomattavasti. Lisäksi digitaalisuus tuo taloushallintoon joustavuutta ja parantaa sen laatua vähentäen samalla virheitä.

Yritykset ja organisaatiot, jotka ovat siirtyneet kohti digitaalista taloushallintoa, ovat saavuttaneet tyypillisesti keskimäärin 30-50 prosentin parannuksen taloushallintonsa tehokkuudessa. Parannusarvioon on laskettu koko prosessi sisältäen taloushallinto-osaston resurssien lisäksi muut työntekijät, jotka osallistuvat prosessiin. Integroidussa taloushallinnossa perustieto sijaitsee vain yhdessä paikassa, jolloin samaa tietoa ei tarvitse käsitellä useaan kertaan. Jopa 90 prosentin tehokkuuden parantuminen on saavutettavissa yksittäisten prosessien kohdalla. Työvoiman, arkistointitilan,

postituksen ja näitä tukevien fasilitteettien kustannussäästöt korostuvat. (Lahti & Salminen 2008)

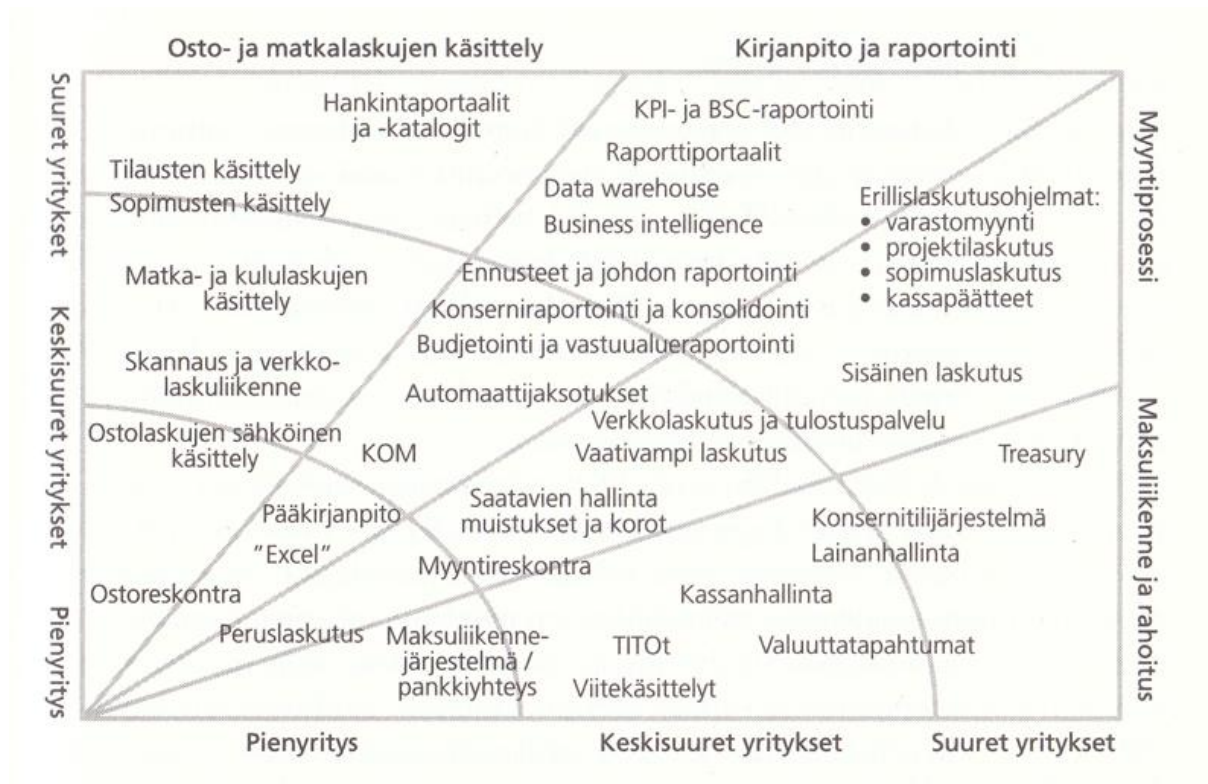
Nykyään digitaalista taloushallintoa voidaan sanoa myös integroiduksi taloushallinnoksi, sillä integraatio ei koske yrityksen sisäisiä järjestelmiä, toimintoja ja työntekijöitä, vaan koko yrityksen arvoketjua (Lahti & Salminen, 2008). Isommassa mittakaavassa integraatioon kuuluvat yrityksen käytössä olevien järjestelmien lisäksi muut yrityksen sidosryhmiin kuuluvien yritysten tai organisaatioiden järjestelmät, jolloin integraatioprojekti kasvaa yritysten väliseksi strategiseksi arkkitehtuurihankkeeksi.

Taloushallintoon olennaisena osana kuuluvat laskut ja niiden myötä laskutus ja laskutukseen liittyvät prosessit. Usein laskujen määrä on suoraan verrannollinen yrityksen kokoon nähden. Yleensä organisaation koon kasvasessa niiden työntekijöiden määrä, joka laskuja käsittelee, kasvaa ja laskun käsittelyprosessit monimutkaistuvat. Toisin sanoen laskunkäsittelyn prosessit monimutkaistuvat.

2.2 Taloushallintojärjestelmät

Yrityksmaailmassa on olemassa järjestelmiä eri yritystoiminnan osa-alueisiin tarpeista riippuen. ERP-ajattelun ydin on tuoda kaikki nämä eri järjestelmät yhdeksi kokonaisuudeksi yhdelle tietokoneelle käyttäen yhtä tietokantaa. Näin yrityksen eri osastot pystyvät käyttämään reaaliajassa olevaa tietoa hyväkseen ja yrityksen sisäisten järjestelmäintegraatioiden määrä ja tarve pienenee johtaen pienempiin kustannuksiin ja tehokkaampaan ajankäyttöön.

Vaikka ERP-ajattelu on kehittynyt siihen suuntaan, että kaikki yritystoiminnassa tarvittavat järjestelmät rakennetaan moduuleina yhdeksi suureksi järjestelmäksi, on yhä olemassa myös erillisiä yrityksen taloushallintoon suunniteltuja ohjelmistoja. Nykyään suurilla yrityksillä sekä organisaatioilla on käytössään jokin markkinoilta löytyvä toiminnanohjausjärjestelmä ja vaikka taloushallinto on laissa säädetty ja näin ollen vakioitua, voivat yritysten tarpeet ja toimiala vaatia hyvinkin erilaisia asioita taloushallintoon liittyen. Yleensä taloushallintojärjestelmän ominaisuudet ja toiminnot määräytyvät yrityksen suuruden mukaan (Kuva 2).



KUVA 2 Yrityksen koko määrittelee sen taloushallintojärjestelmätarpeet (Salmi ja Lahtinen 2008, 35)

2.3 Taloushallinnon prosessit

Taloushallinnoksi kutsutaan järjestelmää, jonka avulla organisaatio seuraa taloudellisia tapahtumia. Taloushallinnon tapahtumia seuraamalla organisaatio raportoi toiminnastaan sidosryhmilleen ja viranomaisille (Lahti & Salminen 2008, 14). Tarkoitukseni ei ole käydä läpi perinpohjin taloushallintoa, kaikkia sen osa-alueita ja niihin liittyviä prosesseja, vaan tarkoitukseni on käydä läpi ne eri taloushallinnon prosessit, jotka liittyvät osto-, matka- ja kululaskuihin ja näin ollen myös näihin järjestelmiin, jotka liittyvät tässä opinnäytetyössä käsiteltävään järjestelmäintegraatioon.

On kuitenkin hyvä kertoa yrityksen taloushallintoon liittyvistä prosesseista lyhyesti, jotta olisi helpompi muodostaa kuva siitä, millaiseen kokonaisuuteen nämä tarkemmin myöhemmin kuvailemani prosessit kuuluvat. Taloushallinto terminä käsitetään yleensä vain laskentatoimeksi, mutta todellisuudessa se on laajempi kokonaisuus ja järjestelmä kuin pelkkä laskentatoimi. Taloushallinto voidaan määritellä

tietojärjestelmien näkökulmasta järjestelmäksi, joka koostuu toisiinsa liittyvistä komponenteista, jotka yhdessä toimiessaan pyrkivät saavuttamaan tietyn tuloksen. Tällainen tulos voi olla esimerkiksi kuukauden tulosraportti tai toimittajalta saapuva ostolasku. (Lahti & Salminen, 2008, 14–16) Usein yritysten ja organisaatioiden taloulshallintotyöt on jettu seuraaviin osa-alueisiin:

- ostolaskuprosessi
- myyntilaskuprosessi
- matka- ja kululaskuprosessi
- maksuliikenne ja kassanhallinta
- käyttöomaisuuskirjanpito
- pääkirjanpito
- raportointiprosessi
- arkistointi
- kontrollit.

Näistä osto-, myynti-, kulu- ja matkalaskuprosesseilla on selkeästi alku- ja loppuvaiheensa sekä näiden välillä selkeästi erottuvat välivaiheet.

2.4 Sähköinen lasku ja ostolaskuprosessi

Sähköinen lasku, jota yleisesti nykyään myös verkkolaskuksi kutsutaan, on sähköisessä muodossa lähetettävä tai vastaanotettava lasku. Siitä löytyvät kaikki ne tiedot, jotka perinteisestäkin laskusta löytyy. Sähköinen lasku voidaan sekä lähettää että vastaanottaa joko datatiedostona (yleensä XML-dokumenttina) tai kuvana. Kuvana lasku välitetään yleensä pdf-dokumenttina tai yleisesti käytössä olevana kuvaformaattina (kuten .jpg tai .tiff). Kuva on yleensä saatu skannaamalla laskun paperiversio jollakin tähän tarkoitettuun ohjelmistolla.

Tilastokeskuksen mukaan (liite 1) keväällä 2011 tehdyn selvityksen mukaan suomalaisista yrityksistä ja organisaatioista jo 79 % eli lähes neljä viidestä on ottanut vastaan sähköisen laskun. Samaa taulukkoa tutkimalla on havaittavissa, että yrityksen koko on suoraan verrannollinen siihen, kuinka yleistä sähköisen laskun vastaanottaminen yritysmaailmassa nykyään on. Sähköisen laskun lähettäneitä sen

sijaan on vähemmän, eli 66 %. Kuitenkin kaksi kolmesta yrityksestä on ainakin kerran lähettänyt laskun sähköisessä muodossa.

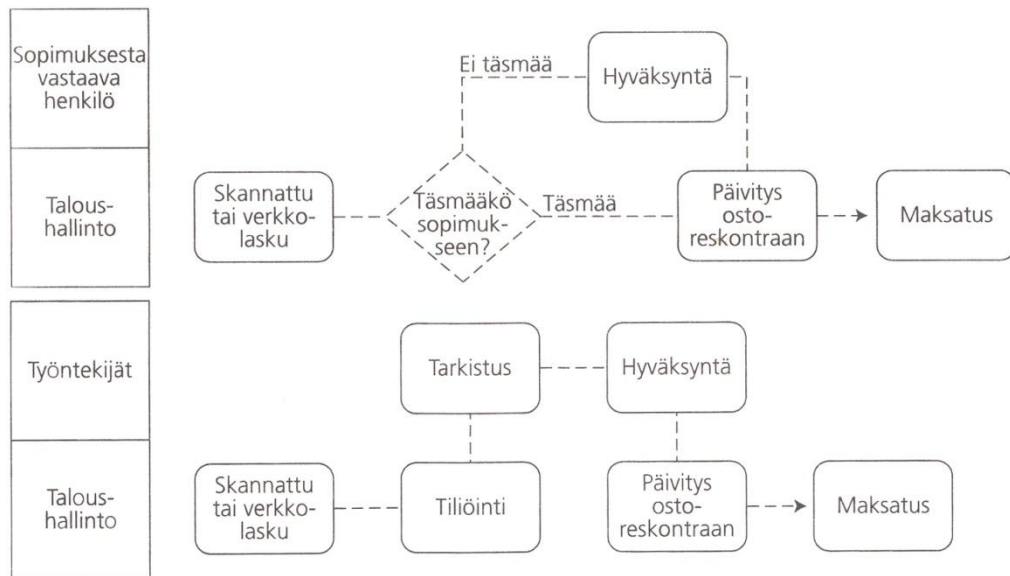
Suomi on muiden pohjoismaiden ohella maailman kärkisijoilla sähköisen laskun käytön suhteen ja kehitys on yhä kohti digitaalista taloushallintoa ja paperitonta laskutusta. Yritykset ovat olleet näyttämässä suuntaa kuluttajien innon ottaessa vasta ensimmäisiä askeleitaan.

Ostolaskuprosessi

Taloushallinnon näkökulmasta ostolaskuprosessi käynnistyy ostolaskun vastaanottamisesta ja päättyy siihen, kun se on maksettu, lisätty kirjanpitoon ja arkistoitu. Isommassa mittakaavassa ostolaskuprosessi alkaa kuitenkin jo ostoehdotuksesta tai –tilauksesta. Tähän väliin prosessia luonnollisesti ehdotuksen ja sitä seuraavan tilauksen hyväksyminen ja tämän jälkeen tavarantoimitus.

Keskityn kuitenkin ostolaskuprosessiin taloushallinnon näkökulmasta jättäen edellä mainitut ostotilaukseen ja toimitukseen liittyvät vaiheet pois. On kuitenkin hyvä tässä vaiheessa mainita, että ostoehdotus ja –tilaus tapahtuvat yleensä tuotannonohjausjärjestelmissä. Sähköinen ostolaskuprosessi taloushallinnon näkökulmasta noudattelee Lahden ja Salmisen mukaan (2008, 49) yleensä seuraavanlaisia vaiheita:

1. Tilaus- ja toimitusprosessi
2. ostolaskun vastaanotto
3. ostolaskun tiliöinti ja kierrätys
4. ostolaskun tarkistus, hyväksyntä ja päivitys ostoreskontraan
5. maksatus
6. täsmäykset ja jaksotukset
7. arkistointi.



KUVA 3. Ostolaskuprosessi (Lahti & Salminen 2005)

Sähköisessä ostolaskuprosessissa perinteinen paperilasku skannataan. Kuvassa 3 ylempänä on kuvattu ostolaskuprosessi, johon liittyy sopimus tai ostotilaus. Ostolasku täsmätään rivitasolla järjestelmästä löytyvään sopimukseen tai ostolaskuun. Tämän jälkeen seuraavan hyväksynnän jälkeen lasku kirjataan ostoreskontraan ja maksetaan. Mikäli ostolaskuun ei liity järjestelmästä löytyvää sopimusta tai ostolaskua, täsmäystä ei tarvita, kuten (kuvan 3 alempi kaavio).

Ennen perinteinen ostolaskuprosessi on ollut huomattavasti monimutkaisempi. Fyysisen laskun kierrättäminen on vaatinut huomattavasti enemmän aikaa ja resursseja. Ilman paperisen laskun skannausta ostolaskuprosessi on noudattanut yleensä seuraavanlaisia vaiheita:

1. Ostolasku saapuu paperilla
2. Lasku viedään tai lähetetään postitse asiattarkastajalle
3. Asiatarkastaja tekee laskulle hyväksymismerkin
4. Asiatarkastaja vie tai lähettää postitse laskun hyväksyjälle.
5. Hyväksyjä tekee laskulle hyväksymismerkin.
6. Hyväksyjä vie tai lähettää postitse laskun ostoreskontrahoitajalle
7. Ostoreskontrahoitaja tallentaa manuaalisesti laskun perustiedot sekä tilöinnin ostoreskontraan
8. Ostoreskontrahoitaja arkistoi paperilaskun mappiin.

9. Ostolaskuista muodostetaan maksuaineisto, joka siirretään pankkiin.

Perinteisen ostolaskuprosessin ongelmana on sen hitaus, aineistojen, kuten laskujen, häviäminen ja sen tarkastelun hitaus jälkikäteen.

MediusFlow on ostolaskujen tilointiin, kierrätykseen ja hyväksyntään kehitetty sovellus. Tästä sovelluksesta kerrotaan lisää myöhemmin kappaleessa 3.

2.5 Kulu- ja matkalaskuprosessi

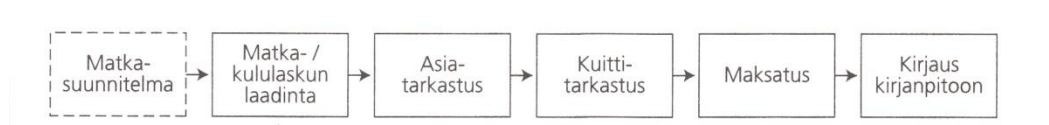
Käytännössä matka- ja kululaskuprosessi aiheutuu yrityksen tai organisaation työntekijän matkustaessa tai ollessa oikeutettu saamaan matkakulukorvauksia, tai yrityksen työntekijä synnyttää yritykselle kulutapahtumia tekemällä pienhankintoja itse. (Lahti & Salminen 2008, 93.)

Laki määrittelee Suomessa tiettyjä verovapaita etuuksia yms, jotka verohallinto vahvistaa vuosittain. Esimerkiksi työmatkat ovat verovapaata tiettyynajaan asti. Työmatkoihin liittyy yleensä myös muita työntekijän maksamia matkakuluja, kuten hotelliyöpymiset tai vastaavat majoituskulut, matkaliput sekä neuvottelukulut, jotka maksetaan kulukorvauksina takaisin työntekijälle. Tässä opinnäytetyössä oleellinen tekijä tätä prosessia on kulu- ja matkalaskuprosessi, johon sisältyy yleensä tietyt vaiheet.

Yleensä kulu- ja matkalaskuprosessi lähtee matkasuunnitelman laatimisesta, jonka työnantaja tai muu suunnitelman hyväksymisestä vastuussa oleva henkilö hyväksyy. Tämän ja varsinaisen työmatkan jälkeen vuorossa on itse laskun laadinta, asiataarkastus, kuittitarkastus, maksatus ja kirjaus kirjanpitoon. Kuten ostolaskuprosessin, niin myös kulu- ja matkalaskuprosessin sähköistäminen nopeuttaa itse prosessia sekä vähentää paperin että työn määrää, jota prosessiin liittyy. Lahden ja Salmisen mukaan (2008, 95) tämä edellä mainittu prosessi jää usein vähälle huomiolle suurissakin organisaatioissa.

Parhaimmillaan matkalaskuprosessi nivoutuu osaksi matkustuksen kokonaisuutta, kun tavoitteena on vähentää ja hallita paremmin myös suoria matkakustannuksia

(lentoliput, hotellit jne.). Kulujen seuraaminen parantuu huomattavasti selkeällä ja keskitetyllä (talous)hallinnolla mikä taas tehostaa kulujen seuranta ja auttaa suuria organisaatioita saavuttamaan huomattavia säästöjä kilpailuttaessaan esimerkiksi lentoyhtiöitä.



KUVA 4. Matka- ja kululaskuprosessi (Lahti & Salminen 2008, 94)

MediusFlow:n avulla voidaan hoitaa myös kulu- ja miksi ei myös matkalaskujen käsittelyyn kuuluvat tiliöinti, kierrätys ja hyväksyntä. Prosessi on yleensä hieman ostolaskuprosessia yksinkertaisempi, sillä laskuun harvoin liittyy ostotilausta. Tällöin laskua ei tarvitse täsmäyttää olemassaolevaan ostotilaukseen.

3 TOIMINNANOHJAUSJÄRJESTELMÄT

Järjestelmiä on nykyään olemassa lähestulkoon kaikkiin yritystoiminnan tarpeisiin. Tarkoitukseni ei ole käydä läpi kaikkia yritysmaailman sovelluksia kattavasti vaan kertoa niistä yleisesti, millaisia järjestelmiä on olemassa ja enemmän niistä järjestelmistä, jotka koskettavat integraatioprojektia ja niiden järjestelmien rajapintoja, jotka olen toteuttanut omissa töissäni ja jotka liittyvät tähän opinnäytetyöhön

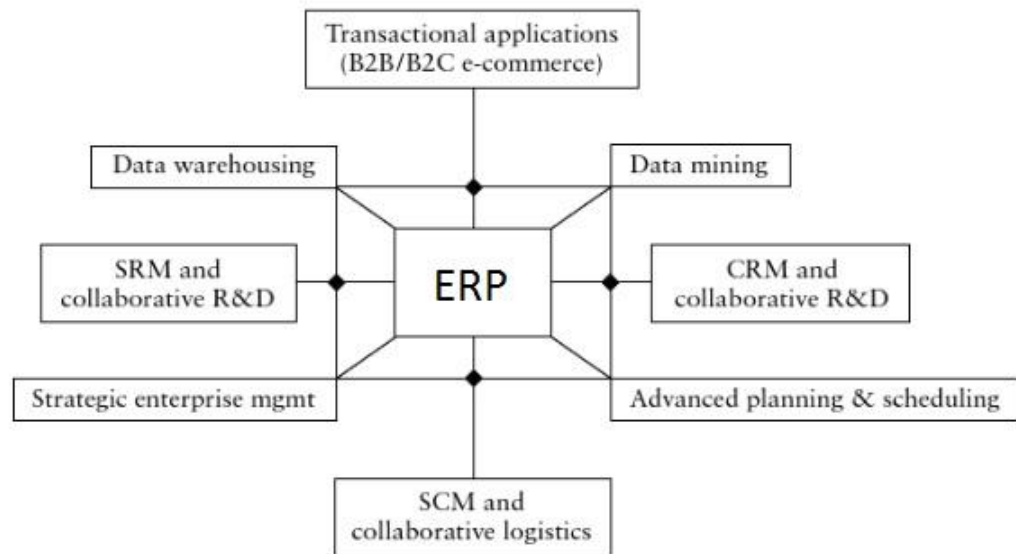
Toiminnanohjausjärjestelmät muodostavat nykyään monien yritysten infrastruktuurit. Yritysmaailman järjestelminä ne käsittävät ja tukevat yritystoiminnan tärkeimpiä toimintoja, kuten hankinnat, tuotanto, myynti, kirjanpito, kulujen hallinta ja henkilöstön hallinta. Toiminnanohjausjärjestelmien periaatteen juuret johtavat tuotannonohjausjärjestelmien kautta aina materiaalien hallintajärjestelmiin. (Themistocleus 2007).

Englanninkielinen termi ”Enterprise resource planning johtaa hieman harhaan. Planning kääntyy suomen kielellä sanaan suunnittelu. ERP-järjestelmät toki antavat työkalut myöskin suunnitella liiketoiminnan eri osa-alueita. Suomen kielessä termi

toiminnanohjaus kuvaa näitä järjestelmiä kuitenkin vielä paremmin; toiminnanohjausjärjestelmillä kyetään reaaliajassa ohjaamaan yrityksen toimintoja.

ERP-järjestelmien käyttöönotolla on monia etuja. Oletettavasti ERP-järjestelmän käyttöönoton on onnistuttava ja sen tulee olla otettu käyttöön, jotta kaikki olemassa olevat hyödyt ovat yrityksen käytettävissä. Suoria etuja ovat toimintojen reaaliaikaistuminen ja sen myötä tehokkuuden paraneminen. Reaaliaikaisuuden ansiosta yrityksen päättäjät pystyvät reagoimaan nopeammin muuttuviin olosuhteisiin ja koon mukaan kasvavien prosessien ja niiden monimutkaisuuden hallinta helpottuu. Edellä mainitut puolestaan nopeuttavat vastaus- ja vasteaikaa asiakkaiden kyselyihin ja tarpeisiin. Epäsuoria etuja ovat yrityksen parantunut maine, parempi asiakastytyväisyys jne. (Bendoly 2008, 13)

Yksi olennaisimmista ja tärkeimmistä hyödyistä on myös tietojen keskittäminen yhteen yhtenäiseen järjestelmäkokonaisuuteen ja tietokantaan. Useimmiten yritystoiminnan tueksi kehitetyt järjestelmät ovat olleet yksipuolisia ja tukivat vain niitä tarkoituksia, joihin ne oli kehitetty. Tällaisia järjestelmiä ei myöskään olla tehty integroitavaksi ulkoisiin järjestelmiin, joten niiden integroiminen ulkoisiin järjestelmiin ei ole ollut välttämättä taloudellisesti ja ajallisesti tehokasta kuin kaikkein suurimmille yrityksille ja organisaatioille. ERP-idean ydin on poistaa tämä hajoavaisuus ja tuoda nämä eri tarkoituksiin kehitetyt järjestelmät yhden kokonaisuuden alle omina moduuleinaan. ERP-järjestelmä koostuu useista eri moduuleista (kuva 5). Moduulit ovat yleensä tähän mennessä olleet itsenäisiä järjestelmiä tai joukko moduuleita ovat muodostaneet kokonaisuutena pienemmän järjestelmän.

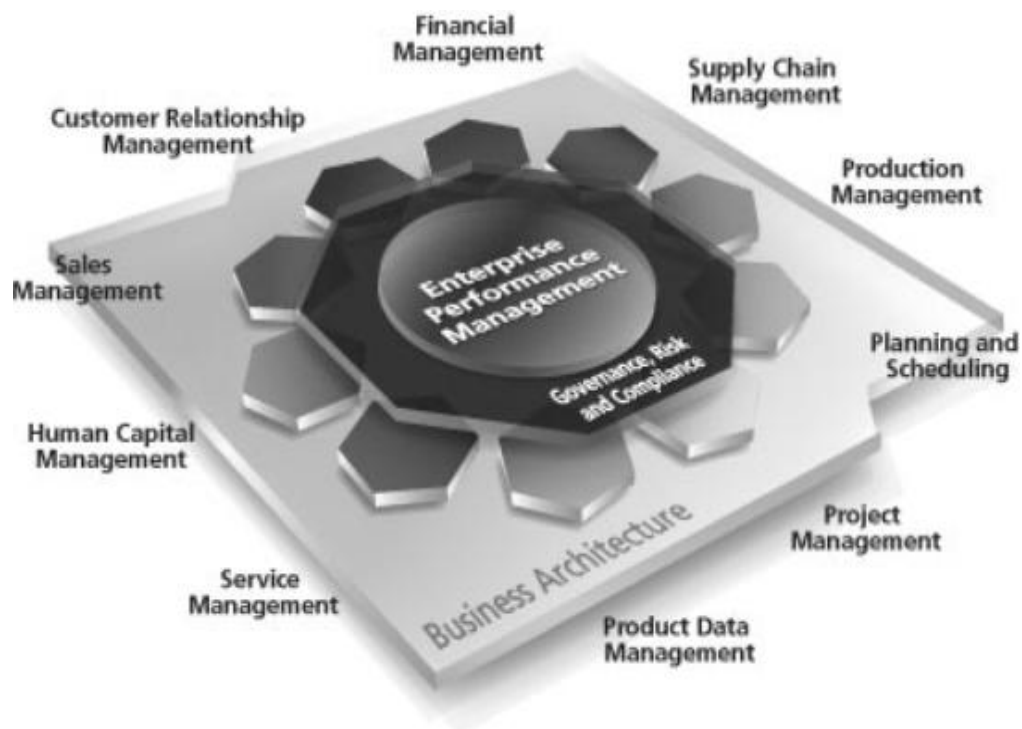


KUVA 5. ERP-järjestelmien muodostuminen ja periaate (Bendoly 2008)

Epicor 9

Yhdysvaltalaisen Epicorin mukaan Epicor 9 on seuraavan sukupolven toiminnanohjausjärjestelmä. Epicor tarjoaa yhdeksi kokonaisuudeksi integroitua pakettia yritystoiminnan tarpeisiin. Se yhdistää Epicorin aikaisemmat tuotteet kuten Epicor Vantage, Vista, ja Scala. Lisäksi siihen kuuluu osia Epicorin Enterprisesta.

Erillisistä moduuleista koostuva Epicor 9 sisältää ainakin kaikki yleisimmät yritystoiminnassa tarvittavat sovellustyypit kuten asiakkuudenhallinta (CRM eli customer relationship management), jakeluketjun hallinta (SCM eli supply chain management), henkilöstönhallinta (HCM eli human capital management), tuotannonhallinta jne. Nämä kokonaisuutena muodostavat siis Epicor 9:n liiketoiminta-arkkitehtuurin (kuva 6).



KUVA 6. Epicor 9 muodostuu eri moduuleista (Epicor 2012)

Epicorin mukaan Epicor 9 skaalautuu yrityksen koon mukaan ja yritys voi valita moduuleista omaan liiketoimintaansa sopivimmat osat. Vaikka E9 sisältää myös yrityksen taloushallintoon keskeisesti kuuluvat osat, kuten pääkirjanpito ja laskutustoiminnot, puuttuu siitä osto- ja laskujen käsittelyyn kuuluvat kierrätystoiminnot organisaation sisällä. Kyseiset prosessit on kuvaattu aiemmin tässä työssä kappaleissa 2.5 ja 2.6.

MediusFlow

Edellisessä kappaleessa kuvatus puutteen vuoksi yritykset joutuvat siis turvautumaan johonkin muuhun järjestelmään, mikäli se haluaa siirtää myös laskun kierrätysprosessin sähköiseen muotoon. Markkinoilta löytyy lukuisia ostolaskusovelluksia, jotka tarjoavat mahdollisuuden esimerkiksi kululaskujen tiliöintiin, kierrätykseen ja hyväksyntään. Yksi tällaisista sovelluksista on ruotsalaisen Mediuksen kehittämä MediusFlow.

MediusFlow:n tarjoamalla sähköisellä laskujen kierrätyksellä tavoitellaan prosessien nopeuttamista ja kustannustehokkuuden parantamista. Osto- ja kululaskun saavuttua lasku skannataan. Skannattu lasku muunnetaan kuvatiedostoksi tai pdf-dokumentiksi. Tämän jälkeen tekstintunnistusohjelma tulkitsee tiedot ja muodostaa niiden perusteella laskun. Tässä opinnäytetyössäni en käy läpi tekstintunnistusohjelmia tarkemmin, koska niiden kanssa työskentely ei kuulunut varsinaisiin tehtäviini integraatioprojektissa.

4 JÄRJESTELMÄINTEGRAATIOT

Integraatio tarkoittaa yksinkertaistettuna vähintään kahden eri järjestelmän välisen tiedonsiirron mahdollistamista eli ts. kahden tai useamman järjestelmän liittämistä toisiinsa siten, että ne toimivat yhtenä kokonaisuutena. Integraatioiden tarvetta on ajanut koko ajan kasvava tiedon määrä ja tarve sen liikkumiselle verkossa esteettä. Nimenomaan jo olemassa olevien järjestelmien ja niiden sisältämän tiedon tuominen uusien tai kehitteillä olevien sovellusten käytettäväksi on johtanut integraatiotekniikoiden kehittymiseen huimaa vauhtia. (Linthicum 2004).

Sovellus-/järjestelmäintegrointi on strateginen lähestymistapa useamman järjestelmän toisiinsa sitomiseen. Usean eri järjestelmän yhteistoiminta sulavasti reaaliajassa verkon yli tarjoavat yrityksille selkeän strategisen edun muihin verrattuna; mahdollisuus harjoittaa liiketoimintaa reaaliajassa maailmassa, jossa täytyy reagoida nopeasti tapahtumiin, vailla viivettä, suo huomattavan edun kilpailijoihin.

Linthicum kuitenkin muistuttaa (2004), ettei järjestelmien integrointi ei ole mikään uusi asia. Niin kauan, kun on ollut olemassa useampi kuin yksi yrityksille suunnattu sovellus tai järjestelmä, on ollut olemassa tarve niiden sitomisesta toisiinsa niin, että ne toimivat keskenään vaihtaen tietoa.

Järjestelmäintegraatio on käsite tai ajattelutapa. Se ei ole tuote tai teknologia. Järjestelmäintegraation avulla voidaan hahmottaa yrityksen tietoteknistä arkkitehtuuria. Se on kokoelma toimintatapoja, joiden avulla yritysten käytössä olevista tietoteknisistä järjestelmistä saadaan irti mahdollisimman paljon. Näin niiden yrityksille tarjoama hyöty on mahdollisimman suuri. (Tähtinen 2005)

4.1 Haasteet

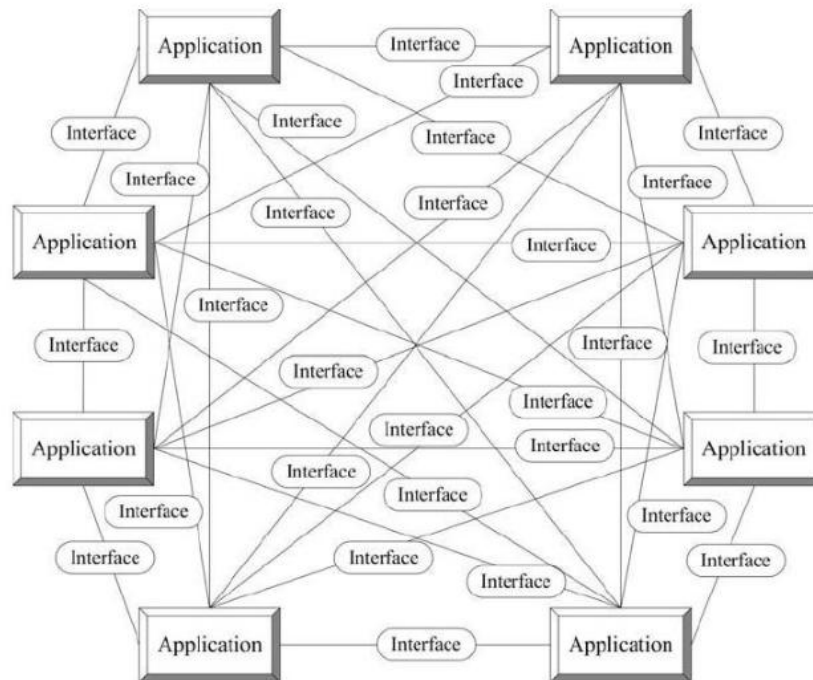
Tietotekniikka kehittyi vauhtia. Kehityksen vauhti itsensä luo omat haasteensa järjestelmäintegraatioille. Nykyiset käytössä olevat tietojärjestelmät on kehitetty eri aikakausina, eri kehitystiimien voimin ja erilaisilla tekniikoilla.

Useimmissa tapauksissa järjestelmät, jotka on kehitetty viimeisintä teknologiaa hyödyntäen vanhentuvat ohikiitävässä hetkessä. Tämän vuoksi järjestelmien integrointi toisiinsa erittäin haastavaa. Lisäksi suuri osa järjestelmiä on saatettu kehittää aikana, jolloin ei edes osattu ajatella, että niiden tulisi kyetä kommunikoimaan toisten järjestelmien kanssa. (Maneuverer & Menard 2010)

Ääriesimerkkinä oletetaan, että kaikki tietojärjestelmään kuuluvat ja sen muodostavat sovellukset tulisi integroida toisiinsa siten, että ne pystyisivät kommunikoimaan kahteen suuntaan (tietoa luetaan sisään ja sitä lähetetään muille järjestelmille). Tästä Maneuvererin ja Menardin mukaan muodostuisi seuraavanlainen yhtälö:

$$i = n(n-1) / 2$$

missä i on rajapintojen lukumäärä ja n on sovellusten lukumäärä. Tämän perusteella kuuden sovelluksen toisiinsa liittämiseen vaadittaisiin 15 rajapintaa. 150 sovelluksen integraatioon vaadittaisiin näin ollen jo 11 175 rajapintaa. Yleensä rajapintojen määrä sovellusten välillä voi olla tätäkin suurempi. Näin ollen päädyttäisiin äärimmäisen sekavaan ja monimutkaiseen integraatioon, jossa pieni muutos voi vaatia valtavan määrän työtä ja aikaa (kuva 7).



KUVA 7. Sovellusten välisten rajapintojen muodostama "verkko" (Maneuvrer & Menard 2010)

Järjestelmien erot muodostavat myös haasteen integraatioille. Yleensä ja lähes poikkeuksetta eri järjestelmien arkkitehtuurit poikkeavat jollakin tasolla toisistaan. Varsinkin järjestelmien tietokannat, joihin tietoa tallennetaan, eroavat toisistaan skeemojensa osalta. Tämän vuoksi tiedon välittämisen lisäksi syntyy tarve myös muuntaa järjestelmän A tietokannasta haettava tieto sellaiseen muotoon, että se sopii järjestelmän B tietokantaan (Linthicum 2004)

Perinteinen järjestelmäintegraatio sopii hyvin yrityksen sisällä toteutettaviin integraatioihin. Integraatioissa tiedon toimintojen luotettavuus, tietojen samankaltaisuus ja tiedonsiirron sekä transaktioiden eheys ovat tärkeitä tekijöitä. Korhonen (2003) lisäksi mainitsee, että integraatio-sovellukset ja –palvelimet nähtiin varsin joustamattomina ja ne vaativat paljon räätälöintiä. Tämän johdosta integraatiot olivat yleensä kahden järjestelmän välille alusta loppuun saakka suunniteltuja toteutuksia, jotka eivät mahdollistaneet muiden järjestelmien tuomista samaan kokonaisuuteen.

Integraatiotekniikat ja –sovellukset ovat kehittyneet huimaa vauhtia vuosien varrella. Teknisten haasteiden ratkaisemiseksi on pyritty kehittämään yhtä joustavampia ratkaisuja samalla, kun järjestelmät ja niiden toiminnot monimutkaistuvat. Lisäksi järjestelmien ja liiketoimintaan suunnattujen ohjelmistojen määrä, joiden tulisi kyetä vaihtamaan tietoa ja toimintoja keskenään sekä yritysten sisällä että yritysten välillä, kasvaa.

4.2 Palvelukeskeinen arkkitehtuuri (SOA)

Puhuttaessa järjestelmien integroinnista erityisesti yksi termi nousee tänä nykyään esille ylitse muiden. Linthicum (2004, 4) puhuu vielä palvelukeskeisestä sovellusintegraatiosta (service-oriented application integration), mutta nykyään puhutaan yleisesti ottaen aina palvelukeskeisestä arkkitehtuurista (Service-oriented architecture eli SOA). Varsinkaan nykyään ei voida puhua uudesta tavasta lähestyä järjestelmien integrointia, sillä keinoja integroida sovelluksia toisiinsa on ollut jo vuosia.

Pulkkisen (2008) mukaan palvelukeskeinen arkkitehtuuri (soa, service oriented architecture) tarkoittaa yksinkertaisimmillaan sitä, tietotekninen infra(struktuuri) nähdään verkkona, jossa palveluita tuotetaan mahdollisimman tehokkaalla tavalla.

Järjestelmäintegraatiot ovat olennainen osa sähköisen liiketoiminnan sovellusten parissa eikä sovellukset voi olla olemassa itsenäisinä kokonaisuuksina. Sen sijaan niiden tulee jakaa tietoa muiden tietojärjestelmien kanssa niin organisaatioiden sisällä kuin niiden välillä. Pitkään järjestelmät tyytyivät vaihtamaan tietoa keskenään ainoastaan informaation tasolla, eli ne vaihtoivat ainoastaan tietoa keskenään. Tällainen tapa oli yleensä hyvin staattinen, joten tämän vuoksi järjestelmien integrointi keskenään oli työlästä ja eikä kovinkaan kustannustehokasta. SOA tarjoaa yrityksille mahdollisuuden tarjota sekä yleisiä sovelluksen palveluja eli servicejä sekä informaatiota. Päällimmäisenä periaatteena on tarjota sovellukseen sellaisia palveluita, joiden kautta järjestelmään voidaan tuoda ja sieltä voidaan kutsua erilaista dataa yleisten ja standardien rajapintojen kautta.

Vielä vuosikymmen sitten puhuttiin, että tulevaisuuden haaste on saada nämä kyseiset palvelut toimimaan organisaatioiden välillä. Viimeisen yli kymmenen vuoden aikana koko ajan pienemmät yritykset ja organisaatiot ovat ottaneet käyttöön jonkinlaisen järjestelmän sekä sisäisen että ulkoisen toimintansa tehostamiseksi, on integraatiotekniikoita pyritty kehittämään mahdollisimman joustaviksi.

Maneuvrer ja Menard kuvaavat (2010) palvelukeskeistä arkkitehtuuria orkesteriksi. Jokainen service on kuin soittaja, joka hoitaa oman osuutensa. Jokaisen soittajan soittaessa oman osuutensa kunnialla soittajista muodostuu orkesteri. Jotta jokainen soittaja soittaa nuottinsa oikein ja oikeaan aikaan, tarvitaan kapellimestari. Kapellimestarin virkaa palvelukeskeisessä arkkitehtuurissa voisi suomen kielellä sanoa liiketoimintaprosessien hallinnaksi (BPM eli Business Process Management).

Korhosen näkemys (2003) tukee Maneuvrerin (2010) vertausta orkesteriin. Korhosen mukaan tietojärjestelmien yhteensovittaminen jo tuolloin muistutti enemmän orkesteria kuin kahden järjestelmän välistä yhteisesitystä, duettoja.

Epicor 9 on pyritty toteuttamaan palvelukeskeisen arkkitehtuurin mukaisesti siten, että sen toiminnot (liiketoimintaprosessit) on helposti ja joustavasti integroitavissa ulkoisten järjestelmien kanssa.

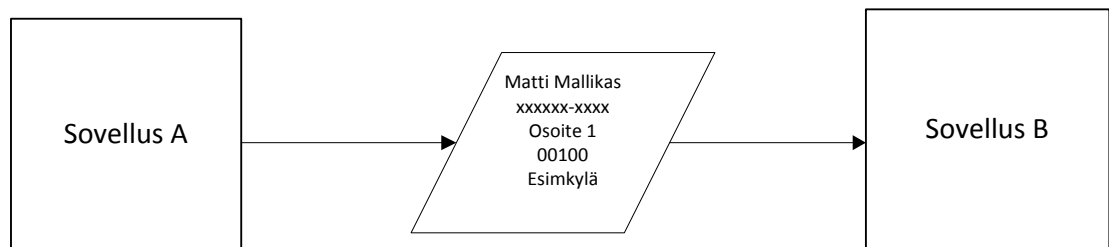
Liiketoimintaprosessien hallinta

Business Process Management (BPM) ryhmittelee joukon liiketoimintaan liittyviä prosesseja yhdeksi kokonaisuudeksi, jotta niitä olisi helpompi optimoida ja hallita. Toisin sanoen BPM tarkoittaa metodien, työkalujen ja palveluiden kokoamista kokonaisuudeksi, jolla voidaan toteuttaa, mallintaa ja optimoida yrityksen tai organisaation arvoketjuun kuuluvia liiketoiminnan prosesseja.

4.3 Informaatiokeskeinen integraatio

Tiedon replikointi

Informaatiokeskeisessä integraatiossa perusajatuksena on, että järjestelmien välinen integraatio tapahtuu suoraan järjestelmien tietokantojen välillä. Yksinkertaisimmillaan tieto luetaan yhdestä tietokannasta ja syötetään samantien toiseen tietokantaan.

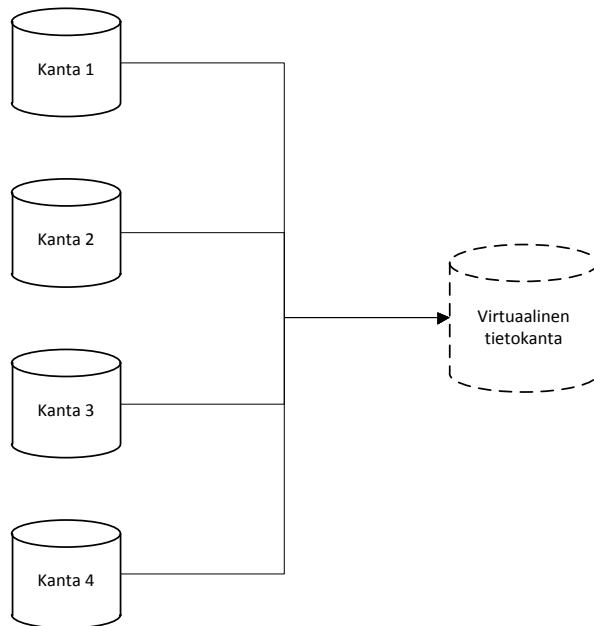


KUVA 8. Informaatiokeskeinen Integraatio (Linthicum 2004)

Informaatiokeskeisessä integraatiossa sovellusten välille on ollut olemassa middleware-ohjelmistoja, jotka lukevat tietoa lähtötietokannasta ja muuntaa sen tarvittaessa määränpäättietokantaan sopivaksi. Tällaista tapaa kutsutaan datareplikoinniksi eli tiedon kopionniksi tai toistamiseksi. Informaatiokeskeisen integraation etu on sen yksinkertaisuus, jonka kautta se on edullinen, helppo ja nopea toteuttaa.

Tietokantojen yhdistäminen

Linthicumin (2004) mukaan data federation (suom. tietojen yhdistäminen) on kahden tai useamman tietokannan yhdistäminen yhtenä kokonaisuutena käsitettäväksi tietokannaksi. Yleensä kaksi tai useampi fyysinen tietokanta yhdistetään virtuaaliseksi tietokannaksi (kuva 9).

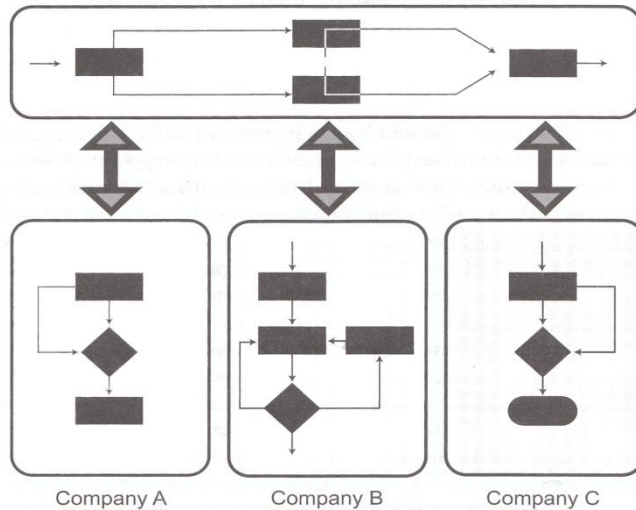


KUVA 9. Yksi tai useampi tietokanta muodostaa yhden, virtuaalisen tietokannan (Linthicum 2004)

Informaatiokeskeiset tietokantaintegraatiot ovat suhteellisen helppo, nopea ja kustannustehokas tapa toteuttaa järjestelmien välinen integraatio. Sen ongelmana on kuitenkin sen joustamattomuus, lyhytikäisyys ja se ei ota huomioon muita järjestelmiä. Nykyään tiedon tulee liikkua myös yritysten keskenään muodostamien sidosryhmien kesken, jolloin keskenään integroitavia järjestelmiä on lukuisia.

4.4 liiketoimintaprosessien integraatio

Linthicumin (2004) mukaan yritysarkkitehtuurin perusidea on tarjota kokoelma helposti määriteltäviä ja keskitetysti hallittavissa olevia toimintoja jo olemassa toimintojen päällä. Nämä jo olemassa olevat toiminnot ovat jo valmiiksi osa jotakin yritystoimintaan suunnattua ohjelmistoa (kuva 10).



KUVA 10. Liiketoimintaprosessien integraatio (Linthicum 2004)

Epicor Service Connectin voidaan katsoa osittain pyrkivän myös liiketoimintaprosessien integraatioon. Joustavasti ja vapaasti luotavat prosessit voidaan toteuttaa yritysten tarpeiden mukaisesti.

4.5 Web Servicet

Nykyään sovellusintegraatioiden ja palvelukeskeisen arkkitehtuuriajattelun yhteydessä puhutaan lähes poikkeuksetta Web Serviceistä. Vaikka käsite suomennetaan vapaasti verkkopalveluksi, kyseessä ei ole perinteinen verkkopalvelu, kuten alaa vähemmän tuntevat voivat helposti luulla. Web service on sovelluksen osa, joka tarjoaa omia toimintojaan muiden sovelluksien käytettäväksi. Koska kunnollista suomenkielistä vastinetta ei mielestäni ole tänäkään päivänä olemassa, puhun vastaisuudessa aina Web serviceistä.

Käsitteenä Web services on muutenkin hieman vaikeasti käsitettävissä. Tähän yksi syy on Talvitien (2004) mukaan jo alkujaan epäonnistunut nimi. Huonosti suomennettavissa oleva alkumuotokin on saanut nykyään kaksi eri versiota: joissakin yhteyksissä 'web services' merkitsee yleisesti web-palveluja (web-sovelluksia) ja 'Web services' näitä uusia SOAP-pohjaisia web-sovelluspalveluja.

Linthicum (2004, 311) kirjoittaa, että Web service –ajattelu oli mullistavin ajattelumallin muutos sitten itse Internetin keksimisen. Hänen mukaansa web servicen

idea on jännittävä: kyky muodostaa fyysisesti eri laitteilla ympärimaailmaa sijaitsevista sovelluksen osista, palveluista, uusia sovelluksia ja kokonaisuuksia, on huikea askel kohti joustavaa tapaa tehdä uusia järjestelmiä. Saman periaatteen mukaisesti web servicet mahdollistavat bisnessovellusten eri osien integroimisen toisiin järjestelmiin huomattavasti joustavammin ja kustannustehokkaamin.

Pulkkisen (2008) mukaan palvelukeskeinen SOA tarkoittaa kahta asiaa, joista ensimmäisenä se voi tarkoittaa Web services –tekniikkaan liittyviä sovellusintegraatioita. Tällöin puhutaan ennenkaikkea http-protokollalla tapahtuvista palvelukutsuista. Hänen mukaansa ”tämä radikaali soa-konsepti muuttaa palveluiden tuottamisen perustekniikoita”.

On hyvä muistaa, ettei SOAI:lla (Service Oriented Application Integration) tai SOA:lla (Service Oriented Architecture) tarkoiteta samaa asiaa kuin SOAP (Simple Object Access Protocol). Kaksi aiempaa mainittua ovat tapoja toteuttaa sovellusten arkkitehtuuri siten, että niiden toiminnot on rakennettu integraatiolähtöisesti. SOAP taas tarkoittaa XML-pohjaista, standardisoitua tietoliikenneprotokollaa, jonka avulla mahdollistetaan verkkopalveluiden etäkutsut.

5 XML

Extensible Markup Language eli XML on merkkauskieli, jonka merkitys on kiistatta merkittävä nykyaikaiselle sovellusten tiedonvaihdon ja sovellusintegraatiolle. XML:n arvo korostuu sen vapaasti määriteltävissä olevassa rakenteessa. XML ei itsessään ole niin merkittävä, vaan merkittäväksi nousevat sen johdannaiset kuten SOAP, WSDL, ebXML ja monet muut XML-pohjaiset tekniikat. (Linthicum 2004, 235.)

XML on alusta alkaen kehitetty nimenomaan tiedonsiirtoon ja –vaihtoon Internetissä ja tietoliikennesyhteyksien avulla, jonka vuoksi se on luonnollinen ja tärkeä väline järjestelmäintegraatioissa. Myös tässä opinnäytetyössä kuvattu integraatio on toteutettu sellaisten sovellusten avulla, jotka käyttävät XML-tekniikoita hyväkseen tiedon välityksessä.

5.1 SGML

Koska XML:stä on kehitetty SGML:stä, eli Standard Generalized Markup Languagesta), on mielestäni hyvä lyhyesti kertoa tästä XML:ää edeltäneestä merkintäkielestä. Kuitenkaan SGML:stä ei kannata kertoa liian yksityiskohtaisesti, koskaeroja XML:ään on loppujen lopuksi niin paljon, että myöhemmin XML:ää opetelessa joutuisi käyttämään aikaa unohtaakseen sen, mitä on oppinut SGML:stä. XML on kuitenkin hyvin selkeästi SGML:n jälkeläinen. Tämän vuoksi SGML tulee nousemaan esille jatkossa, joten on hyvä ymmärtää tai olla vähintään tietoinen vallitsevasta laajemmasta kokonaiskuvasta.

SGML syntyi tarpeesta varastoida dataa yhdestäkään ohjelmistopakettista ja ohjelmistojen myyjästä riippumatta. SGML on metakieli, eli kieli jolla kuvataan merkkauškieliä. HTML on yksi näistä merkkauškielistä, joten HTML:ää voidaan kutsua yhdeksi SGML:n sovellukseksi. HTML:n lisäksi on olemassa jopa satoja merkkauškieliä, jotka ovat kehitetty SGML:n avulla. XML:ssä näitä sovelluksia kutsutaan usein merkkauškieliksi (markup languages). Muista olemassa olevista sovelluksista en tämän opinäytetyön yhteydessä puhu, ellei tarve erikseen niin vaadi. SGML-määrittelyissä esitellään, mitkä merkit tulkitaan dataksi ja mitkä merkkaukseksi. XML:ssä merkkaukseksi on valittu mm. merkit < ja >.

SGML:n määrittelyssä kehittäjä voi identifioida erilaisten sääntöjen ja informaatioanalyysien tulosten avulla dokumenttityyppejä. Esimerkkejä dokumenttityypeistä ovat mm. raportti, esite tai tekninen manuaali. Kullekin kehittäjä määrittelee DTD:n, jossa valittuja merkkejä käyttäen voidaan (North & Hermans. 2000, 13)

DTD eli document type definition on koko SGML- (ja XML-) sovelluksen ydin, jonka tasosta onnistuminen riippuu. Siinä määriteltäviä informaatioelementtejä käyttämällä varsinainen tieto merkataan sille sovellukseen eritellyillä tunnisteilla. Mikäli dokumentin tyypin määrittely (DTD) on tehty hätäisesti, se saattaa tarvita jatkuvaa parantelua, muuntelua ja korjailua. Tämä puolestaan ei ole kustannustehokasta, koska joka kerta kun DTD:tä joudutaan muokkaamaan. Jo pienikin muutos voi aiheuttaa sen,

että merkattua informaatiota saatetaan joutua muokkaamaan mahdollisten vidheiden vuoksi.

5.2 XML:n vahvuudet

Merkkauskielten ja varsinkin XML:n käyttö on yleistynyt jatkuvasti. Luonnollisesti jatkuvasti kasvaneeseen suosioon on olemassa lukuisia syitä. XML:ää voidaan käyttää yhdessä olemassa olevien Web-protokollien, kuten HTTP, ja Web mekanismien, kuten URL:t , kanssa ilman, että XML asettaisi yhteiskäytölle lisärajoitteita. Toiminnasta yhdessä verkkoprotokollien kanssa voidaan myös vetää se johtopäätös, että XML on kehitetty Webiä ajatellen. XML:stä on jätetty pois ne ominaisuudet, jotka eivät SGML:ssä soveltuneet niin helposti Webissä käytettäväksi.

XML on nimensä mukaisesti laajennettavissa oleva merkkaukieli. Koska XML-dokumentin rakenne ja elementtien nimet voidaan luoda vapaasti mahdollisimman kuvaaviksi, se on ihmisen luettavissa ja helposti tulkittavissa. Kuva 11 esittää yksinkertaista XML-dokumenttia, joka sisältää tilaustietoja. Tarkemmin tarkasteltuna voidaan huomata, että esimerkin XML-dokumentti sisältää vain yhden tilauksen kaksi eri riviä. Kiitos yhden XML-kuvauskielen vahvuuksista kyseisen dokumentin sisällä voidaan siirtää lähes rajoittamaton määrä eri tilauksia. Tilaukset puolestaan voivat sisältää periaatteessa rajattoman määrän tilausrivejä jne.

```

<?xml version="" encoding="utf-16"?>
<tilaukset>
  <tilaus>
    <tilausrivi>
      <tilausnro>01</tilausnro>
      <asiakas>Asiakas1</asiakas>
      <tilausrivinro>1</tilausrivinro>
      <tuote>tuote</tuote>
      <tilattuKPL>3</tilattuKPL>
      <toimitettuKPL>2</toimitettuKPL>
    </tilausrivi>
    <tilausrivi>
      <tilausnro>01</tilausnro>
      <asiakas>Asiakas1</asiakas>
      <tilausrivinro>2</tilausrivinro>
      <tuote>tuote2</tuote>
      <tilattuKPL>25</tilattuKPL>
      <toimitettuKPL>25</toimitettuKPL>
    </tilausrivi>
  </tilaus>
</tilaukset>

```

KUVA 11. Yksinkertainen XML-dokumentti

XML:n käsittelyyn soveltuvia ohjelmia on nykyään suhteellisen helppo ohjelmoida. XML on HTML:n tavoin pyritty pitämään mahdollisimman yksinkertaisena. Monet SGML:lle ominaisista monimutkaisista ominaisuuksista on jätetty pois. (North & Hermans, 2000).

Kuvassa 13 on kuvattu notepad++:lla luotu esimerkkinä yksinkertainen XML-dokumentti, joka sisältää liiketoimintaan liittyvää tietoa. Tässä tapauksessa kysymyksessä on tilauksen toimitustietoja.

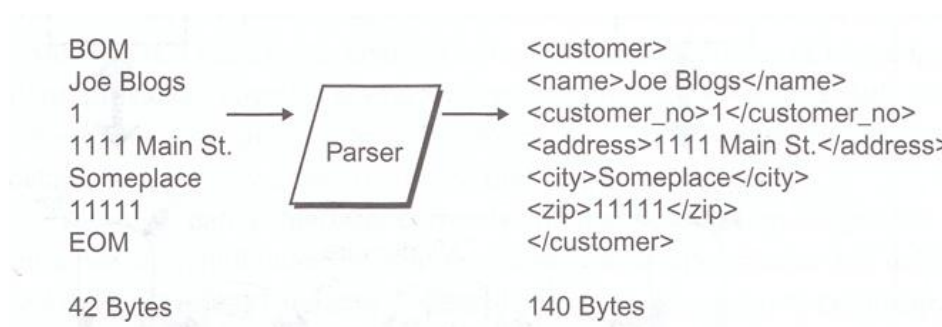
5.3 XML ja sen heikkoudet

Onko XML sitten ratkaisu kaikkiin ongelmiin tuomatta itse lisää ongelmia järjestelmäintegraatioiden maailmaan? Valitettavasti sitä se ei ole. Siinä missä mistä tahansa muusta tekniikasta, myös XML:stä löytyy omat heikkoutensa. XML on tekstimuotoinen tiedostoformaatti, joka siirtää tietoa. Yksi XML:n heikkouksista on se, että se on tekstipohjainen tekniikka, jolloin myös siihen tallennettavassa tiedossa voi olla merkkejä, jotka voivat haitata sen käyttöä tietyissä järjestelmissä. Mikäli esimerkiksi mikäli jokin elementti sisältää yhden hipsun, voi tämä haitata järjestelmää,

joka muodostaa XML:n elementtien sisältä tulevan informaation perusteella SQL:n insert lauseita. XML on myöskin melko raskas.

Myös XML:n vapaamuotoisuus luo oman ongelmansa järjestelmien integroimiseen keskenään, koska järjestelmillä ja niiden tietokannoilla, joita ne käyttävät, on erilainen rakenne ja kolumnien nimet eivät välttämättä täsmää. Tällöin kuvioihin tulevat näiden ongelmien ratkaisemiseen kehitetyt XML skeemat (XML Schema) ja XML – dokumenttien muunnokseen tarkoitetut XSL ja XSLT. XML-tiedostossa olevan datan käsittelyyn tarvitaan tänäkin päivänä sovellus, joka tulkitsee datan ja vie sen tietokantaan. Tämän tyyppisiä sovelluksia kutsutaan myös middlewareksi.

Tekstimuotoisuutensa vuoksi XML-dokumentti voi äkkiä laajeta hyvin suureksi, jolloin tiedoston koko kasvaa. XML-muodossa tiedoston koko voi kasvaa moninkertaiseksi jo pienellä määrällä tietoa (kuva 12). Nykyajan tietoliikenneyhteyksien kasvavan nopeuden ansiosta tämä ei ole välttämättä enää tiedonsiirron kannalta merkittävä asia, koska tiedoston siirtämiseen ei tarvita enää niin paljoa aikaa. XML-tiedoston prosessointiin käytettävä aika voi kuitenkin olla yhä ongelma varsinkin silloin, kun käsiteltävä XML-dokumentti sisältää suuria määriä tietoja tai käsiteltäviä XML-tiedostoja on paljon.



KUVA 12. XML-muodossa tiedoston koko kasvaa äkkiä (Linthicum (2004))

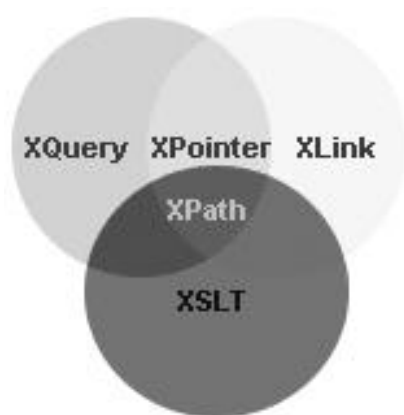
5.4 XPath

XPathia käytetään XML-dokumenttien elementtien ja attribuuttien (ts. XML-dokumentin puurakenteessa) hakuun ja ns. niiden seassa navigoimiseen. XPathissa käytetään standardien mukaisia ilmauksia, jotka muistuttavat huomattavan paljon ilmauksia, joita voi nähdä tavallisen kotikoneen tiedostopoluissa (W3Schools, 2012).

Nämä polut (paths) ilmaisevat siis tietyn noden sijainnin XML-dokumentissa. XPath on IT-alalla laajalti hyväksytty ja W3-konsortion tukema standardi.

XPath sisältää nykyään yli sata sisäänrakennettua funktiota. Näiden funktioiden avulla voi tehdä mm. merkkijono-, numero-, päivämäärä- ja aikavertailuja. Niillä voi manipuloida XML-dokumentin tietoja ja vaikka mitä muuta.

XPath on merkittävä elementti XSLT-standardissa. On lähes mahdotonta tai vähintään hyvin vaikeaa luoda XSLT-dokumentteja ilman XPath-tuntemusta. Tämä johtuu siitä, että sekä XPath- että XLS-tekniikat nojaavat paljon polkumaisiin viittauksiin XML-dokumenttien tiedoissa.



Kuva 13. XPath on eri tekniikoiden yhdistelmä (W3C 2012)

Sujuva työskentely Epicorin Service Connectin työkaluilla edellyttää edes jonkinmoista XPath-tietämystä. Sen täydellistä hallitsemista ei vaadita, mutta käsitteenä on hyvä ymmärtää, mitä XPath on ja mihin sitä yleensä käytetään.

5.5 XML Schema

XML Schema on World Wide Web Consortiumin (W3C) suosittelema standardi XML-dokumenttien rakenteen kuvaamiseen. XML schemoista puhuttaessa se yleensä lyhennetään XSD, joka oletettavasti tulee XML schemojen tiedostopäätteestä XSD. Kuvassa 14 on esimerkkinä aiemmin esitellyn yksinkertaisen XML-dokumentin rakenteen kuvaaminen XML scheman avulla:


```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://Epicor.com/SC/UserSchema"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://Epicor.com/SC/UserSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="tilaus">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="tilausrivi">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="tilausnro" type="xs:unsignedByte" />
              <xs:element name="asiakas" type="xs:string" />
              <xs:element name="tilausrivinro" type="xs:unsignedByte" />
              <xs:element name="tuote" type="xs:string" />
              <xs:element name="tilattuKPL" type="xs:unsignedByte" />
              <xs:element name="toimitettuKPL" type="xs:unsignedByte" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Kuva 14. Yksinkertaisen XML-dokumentin rakenteen kuvaus XSD:n avulla.

Epicor Service Connectin Workflow Designer –työkalun avulla työskentely edellyttää XML skeemojen hallitsemista. Skeemojen avulla XML-dokumentin rakenne voidaan esitellä myös ohjelmille niin, että ne ymmärtävät sen.

5.6 XSLT

XML-peräisiä standardeja on menneiden vuosien aikana viuhunut ympäriinsä ja Linthicumin mukaan (2004, 246) niiden määrä on ns. tulvinut yli äyräiden. On ollut RosettaNettiä, BizTalkia jne. Kuitenkin viime aikoina yksi XML-pohjainen tekniikka on noussut pikkuhiljaa yli muiden. XSLT:n tavoitteena on tarjota standardisoitu mekanismi XML-dokumenttien muuntamiselle.

XSLT on kieli, joka on suunniteltu muuntamaan XML-dokumentti toiseen muuttaen täten sekä tiedon skeeman (muotoilun) ja sisällön (Linthicum 2004, 248). XSLT:tä

voidaan käyttää myös XML:n muuntamiseen muiksi merkinäkieliksi, kuten esimerkiksi HTML:ksi.

Koska lähes poikkeuksetta jokaisella sovelluksella on oma, ainutlaatuinen semantiikkansa, sovellusten välillä liikkuvat dokumentit on muunnettava. Sekä sisällön että sen rakenteen on oltava semantiikaltaan yhteensopivaa, jotta se on vietävissä alkuperäisestä järjestelmästä kohdejärjestelmään. Mikäli näin ei ole, tietojen päivitys mitä todennäköisimmin tulee epäonnistumaan.

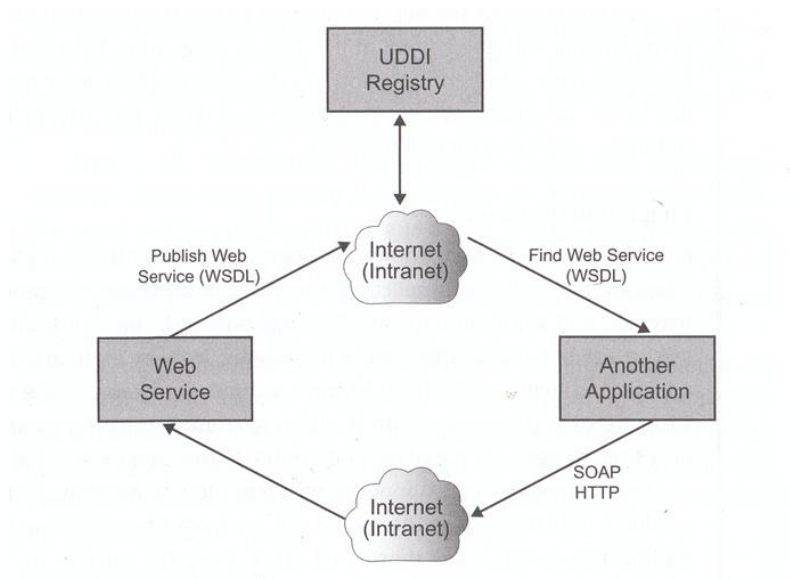
XML-dokumentin skeeman ja sisällön muokkaamisen lisäksi XSLT voi suorittaa myös muita erilaisia tekstinkäsittelyoperaatioita, joihin sisältyy mm. tekstipohjaisten dataformaattien, kuten PDF, muodostaminen.

5.7 Simple Object Access Protocol (SOAP)

SOAP eli Simple Object Access Protocol määrittelee Extensible Markup Language (XML)-pohjaisen sanoma- tai viestiformaatin, jonka avulla Web servicet (sovellusten osat) voivat kommunikoida toistensa kanssa verkon yli. Linthicumin (2004, 313) mukaan Webin heterogeenisen ympäristön vuoksi on tarpeellista, että sovellukset tukevat standardin mukaisia ja yleisesti käytössä olevia sanoma- tai viestiformaatteja. SOAP tarjoaa yhden tällaisen standardin, jonka avulla voidaan kutsua toisen sovelluksen funktioita riippumatta käytössä olevasta laitteistosta, käyttöjärjestelmästä tai ohjelmointikielestä. SOAP tarjoaa lähinnä ohjelmarutiinien etäkutsumista. SOAP eroaa yleisesti Windows-maailmassa käytössä olevasta DCOM-tekniikasta (Distributed Component Object Model), unix-maailman CORBA:sta (Common Object Request Broker Architecture) ja Javan RMI:stä (Remote Method Invocation) lähinnä XML-rakenteensa ja tiiviin internet-kytkentäisyydensä ansiosta.

Talvitie (2004) kertoo, että Simple Object Access Protocol on nimensä osalta toisaalta hieman harhaan johtava. Käännös ”yksinkertaisten olioiden kutsumistekniikka” ei välttämättä kerro SOAP:sta mitään. Olio-sana viittaa harhaanjohtavasti olio-ohjelmointiin, vaikkei olioilla ja web service –tekniikoilla itseasiassa ole juuri sidoksia keskenään. Talvitie (2004) lisää, että ”hajautettujen sovelluspalvelujen ehtona on vakaa ja ennakoitava toimintaympäristö”. SOAP:n

kaltaiset standardit, ts. yhteisesti sovitut käytännöt mahdollistavat ja luovat hajautetulle sovelluskehitykselle ja luovat mahdollisuuksia järjestelmien väliselle integraatiolle.



KUVA 15. SOAP (Linthicum 2004)

Epicor Service Connect käyttää SOAP:ia hyväkseen, jotta muut järjestelmät tai Service Connectiin sijoitettavat toiminnot voivat kutsua Epicor 9:n liiketoimintametodeja.

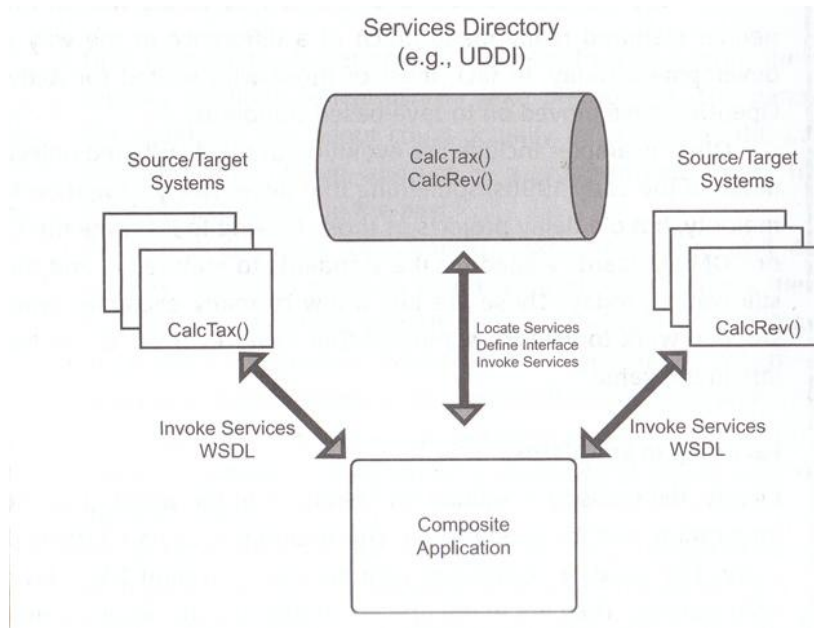
5.8 Web Service Description Language (WSDL)

Web Service Description Language eli WSDL on kokoelma metadataa XML-pohjaisista palveluista, joita käytetään kuvaamaan mitä liiketoimijat tekevät ja kuinka heidän palveluihinsa pääsee käsiksi elektronisesti. WSDL perustuu SOAP:iin, ja se määrittelee sen, kuinka havaita toiminnallista ja teknistä tietoa Web serviceistä internetin välityksellä. (Linthicum, 2004, 316).

WSDL-dokumentti koostuu monesta eri elementistä:

- Tyypinmäärittely dataelementeille. Yleensä tähän käytetään XML Schemoja
- Sanomamäärittely, joka koostuu yhdestä tai useammasta tyypitetystä dataelementistä.

- Operaatioiden määrittelyt, jotka ovat abstrakteja kuvauksia toiminnoista, joita palvelu tukee tai tarjoaa.
- PortType määrittelyt, eli listaus operaatioista, joita Web Service tukee tai tarjoaa.
- Ns. bindausmääritelmät, jotka kuvaavat bindaukset portTypejen ja protokollien välillä (esimerkiksi SOAP tai HTTP GET/POST).
- Palvelukuvaukset, eli lista bindauksista.



Kuva 16. WSDL hoitaa palvelun kuvauksen (Linthicum 2004)

6 INTEGRAATIOSOVELLUKSET

Integraatiosovellukset ovat ohjelmistoja, joiden tarkoituksena on mahdollistaa järjestelmien väliset integraatiot. Usein puhutaan middlewareista tai järeämpien ohjelmistojen kohdalla integraatiopalvelimista. Epicor Service Connect ja Medius Integration gateway ovat integraatiosovelluksia. Tässä kappaleessa tarkoitukseni on kertoa yleisesellä tasolla siitä, mitä nämä sovellukset ovat ja mihin niitä käytetään. Lisäksi kerron yksityiskohtaisemmin Service Connectista ja Medius Integration Gatewaysta.

Middleware

Middleware- tai väliohjelmisto on mekanismi, joka mahdollistaa kahden tai useamman entiteetin välisen (sovellus, järjestelmä tai tietokanta) yhteydenpidon. Toisin sanoen middleware on minkä tahansa tyyppinen ohjelmisto (software), joka helpottaa kahden tai useamman eri ohjelmiston välistä kommunikointia. (Linthicum. 2004, 116).

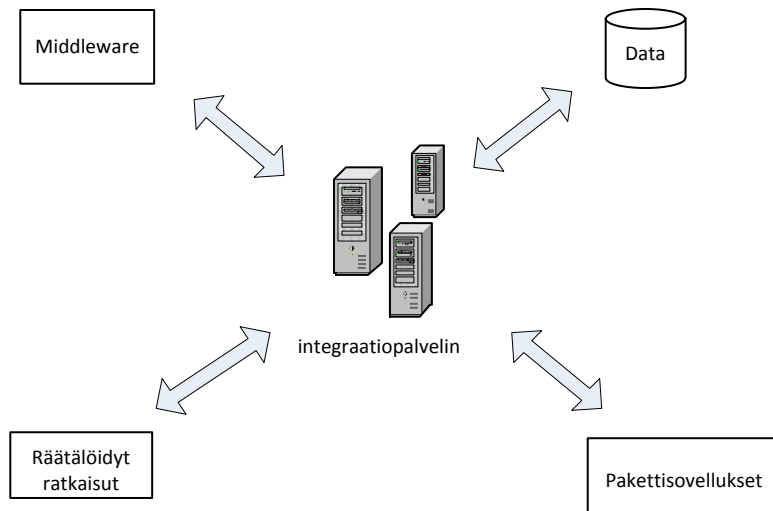
Yllä kuvatun perusteella voi olettaa, että middleware on käsitteenä olennainen osa järjestelmäintegraatioiden maailmassa. Middleware-ohjelmistot pyörivät lähes poikkeuksetta aina palvelinkoneella. Ne toimivat käyttäliittymää hallinnoivan ohjelman (client) ja tiedon tallennuksesta ja säilytyksestä vastaavan ohjelman (SQL) välissä. Middlewarea toisinaan kutsutaan myös sovelluspalvelimiksi.

Middleware on keskeinen käsite tämän opinnäytetyön yhteydessä, koska tämä opinnäytetyö käsittelee kahden järjestelmän, Epicor 9:n ja MediusFlow:n, välistä integraatiota. Integraatiossa toteutetut rajapinnat on toteutettu middleware-ohjelmistojen avulla. Middleware on yleensä järjestelmän osana toimitettu ohjelmisto, joka mahdollistaa tai helpottaa heidän järjestelmän integrointia ulkoisiin järjestelmiin.

Pitkälle kehitettyjen väli- tai integraatio-ohjelmistojen ansiosta teknologioiden ja arkkitehtuurin tuntemuksen tarve vähenee. Omien kokemusteni perusteella rajapintojen rakentaminen järjestelmien välille ei vaatinut tuntenusta teknologioista, joilla esimerkiksi tässä opinnäytetyössä esiteltävän integraation toteutukseen käytetyt integraatiosovellukset on rakennettu.

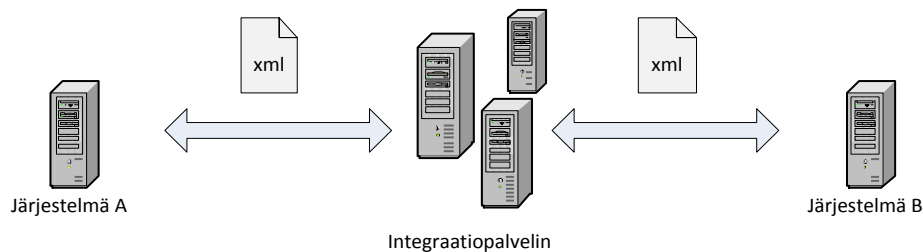
Integraatiopalvelin

Integraatiopalvelimilla tarkoitetaan ohjelmistoja, jotka toimivat tiedon tai kokonaisten palvelujen välittäjinä yhden tai useamman järjestelmän välillä. Ennen kaikkea niiden avulla järjestelmien välinen tiedonvaihto onnistuu riippumatta siitä, miten tieto esitetään tai miten siihen on päästy käsiksi.



KUVA 17. Integraatiopalvelin mahdollistaa integraation eri tyyppisten sovellusten välille (Linthicum 2004)

Yksinkertaistettuna integraatiopalvelin hallinnoi sovellusten välisiä toimintoja. Tiedonvaihto perustuu sanomiin, joita yksi järjestelmä lähettää ja toinen lukee. Järjestelmät voivat olla mitä tahansa järjestelmiä tai sovelluksia. Tämä tuo joustavuutta järjestelmien integroimiseen toisiinsa.



KUVA 18. Integraatiopalvelin hoitaa tiedonvälityksen (Linthicum 2004)

Yleensä integraatioiden yksi haaste on järjestelmien sisältämien tietojen ja semantiikan erot. Tämä voidaan ottaa huomioon integraatiopalvelimien avulla, joissa lähdetietokannasta luettava tieto muunnetaan kohdetietokantaan sopivaksi.

Integraation toteuttamiseen käytettiin Epicorin Service Connect – ja Mediuksen Medius Integration Gateway –sovelluksia. Käytettäviä tekniikoita integraatiossa ovat jo aiemmin käsitellyt XML, XSLT, XPath ja XML Schemat. Sekä Epicor Service Connect että Medius Integration Gateway hyödyntävät näitä tekniikoita, joten

integraatioon kuuluvien rajapintojen toteuttaminen on kohtalaisen helposti toteutettavissa.

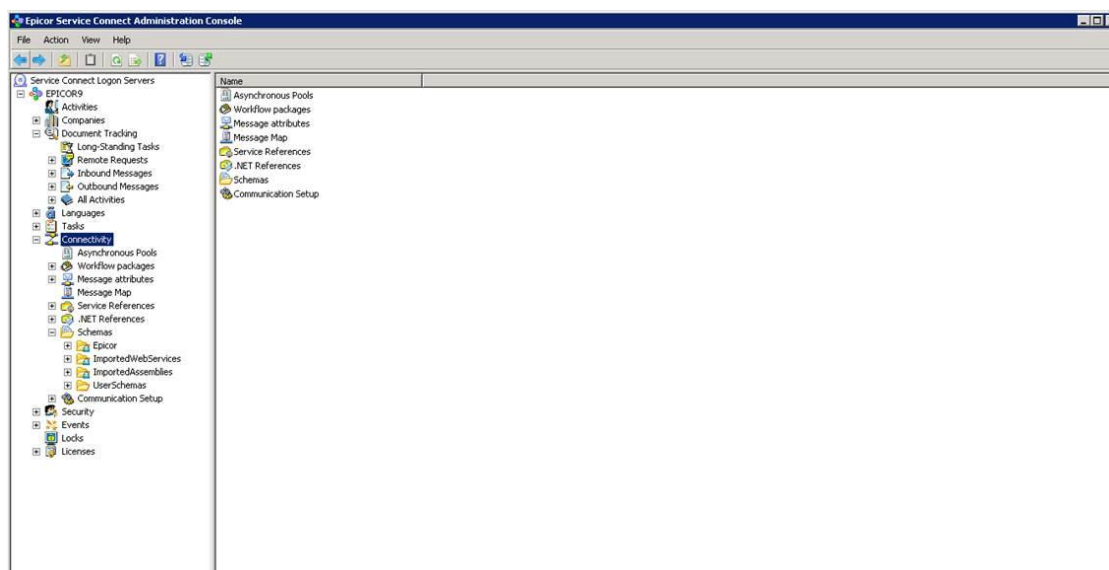
6.1 Epicor Service Connect

Service Connect on Epicorin kehittämä middleware/integraatiopalvelin. Se mahdollistaa ulkoisten järjestelmien integroimisen Epicorin tuotteisiin. Smart Time toteuttaa lähes poikkeuksetta kaikki ulkoisten järjestelmien integraatiot ja niihin kuuluvat rajapinnat Epicor 9:ään Service Connectin ja siihen kuuluvien työkalujen avulla.

Service Connect on kehitetty tukemaan prosesseja ja yhdistämään liiketoiminnan kokonaisuuksia, sovelluksia ja käyttäjiä. Service Connect käyttää avoimia, koko toimialan kattavia standardeja ja teknologioita. Service Connect käyttää XML-pohjaisia tekniikoita tiedonvälityksen välineenä. Sen toteutuksessa on käytetty Microsoftin DNA arkkitehtuuria tukien .NET konseptia.

Epicor Service Connectin hallinnoimiseen käytettävä Service Connect Administration Console on itseasiassa Microsoft Management Console, jonka avulla Service Connectin eri toimintoja voidaan hallinnoida. Se sisältää:

- -Kommunikaattorit (kuuntelijat ja speakerit eli ns. kaiuttimet) ja postaaajat (viestien lähettämiseen), jotka mahdollistavat yhteyden eri protokolliin ja tekniikoihin, kuten Web servicet, COM, FTP, HTTP, e-mail(SMTP, POP3), tiedostot, Microsoft Message Queue, Sonic MQ sekä IBM MQ, perustuen.
- Service Connect Access Serverin (integraatiopalvelin), joka liittää prosessit toisiinsa.
- Workflow Designerin workflow-prosessien rakentamiseen.
- XML Mapperin, jonka avulla tietojen siirtäminen XML-tiedostojen elementtien välillä onnistuu graafisen käyttöliittymän avulla.



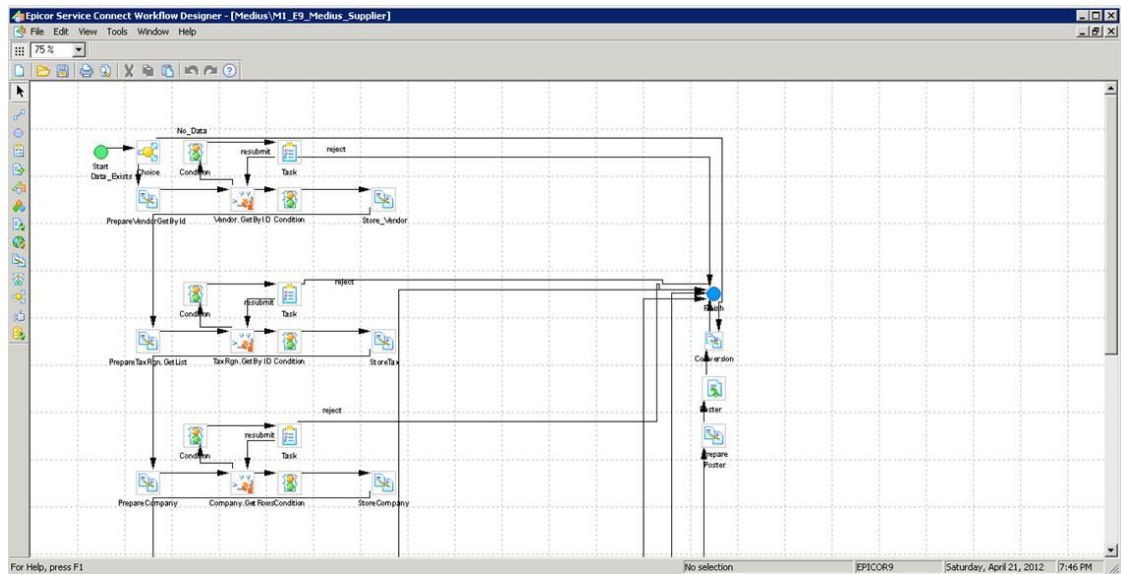
KUVA 19. Epicor Service Connect Administration Console

6.1.1 Workflow Designer

Prosessit (yleensä kutsutaan tässä yhteydessä workflow:ksi) ovat keskeinen osa Service Connectia. Se tarjoaa joustavan välineen liiketoimintalogiikan hallitsemiseen. Näiden workflow-prosessien käyttäminen mahdollistaa eri liiketoimintaprosessien työkulun kuvaamisen ja siihen liittyvien toimintojen automatisoinnin. Esimerkiksi käyttäjä voi suunnitella prosessin, joka lähettää ostotilauksen E9:stä ulkoiseen järjestelmään tai vastaanottaa ja ostolaskun ulkoisesta järjestelmästä.

Workflow prosessi on joukko tietyssä järjestyksessä suoritettavia toimintoja, jotka käsittelevät XML-sanomissa liikkuvaa tietoa. Tärkein tavoite on suorittaa tiettyjä liiketoimintoja. Workflow voi sisältää sekä teknistä tai liiketoimintaan liittyvää tietoa.

Workflow Designerissa prosessit (workflow schemas) kuvataan kaavioina sisältäen sen aikana suoritettavat päätoiminnot ja niiden välillä olevat riippuvuudet. Komponentit kuvaavat yleensä jotakin liiketoimintoa ja viivat niiden välillä sitä, kuinka tietoa sisältävät dokumentit (XML sanomat) liikkuvat näiden komponenttien välillä.



KUVA 20. Workflow Designer

6.1.2 Komponentit Workflow Designerissa

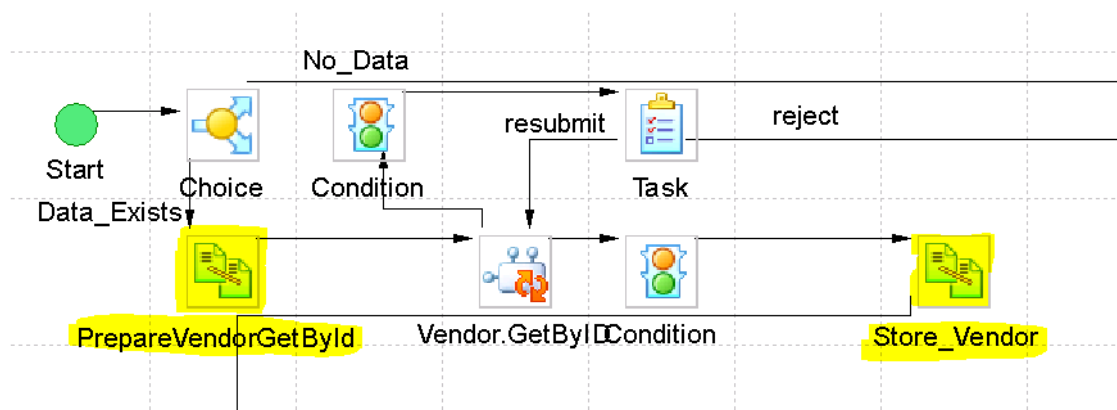
Workflow Designer tarjoaa joukon komponentteja, joiden asetuksia muuttamalla voidaan määritellä miten ne toimivat. Jotta integraation kuvauksessa mahdollisesti esiintyvien kuvankaappausten ymmärtäminen olisi helpompaa, käyn seuraavaksi läpi näiden eri komponenttien käyttötarkoituksia ja säätömahdollisuuksia. Nuolia komponenttien välillä kutsutaan connection-komponenteiksi, jotka kuvaavat tiedon liikkumista workflow-prosessin sisällä ja prosessin etenemisjärjestyksen. Komponenttien välille tulee aina luoda yhteys Connection-komponentilla.

Alku-, lopetus- ja splitter –komponentit

Jokaisessa workflow:ssa on aloitus- ja lopetuselementti. Ne ovat olemassa valmiina jo uutta workflow:ta luotaessa, sillä ne ovat pakollisia. Näistä oleellisempi on Start-komponentti, johon määritellään yleensä prosessiin saapuvan XML-dokumentin rakenne XML-skeematiedoston avulla. Näin ollen prosessissa seuraavana oleva elementti osaa päätellä XML-dokumentin rakenteen. Nämä komponentit ovat selkeästi erotettavissa: start-komponentti on yleensä kaavion vasemmassa reunassa sijaitseva vihreä ympyrä ja finish-komponentti vastaavasti sininen, kaavion oikeassa reunassa sijaitseva komponentti (kuva 21).

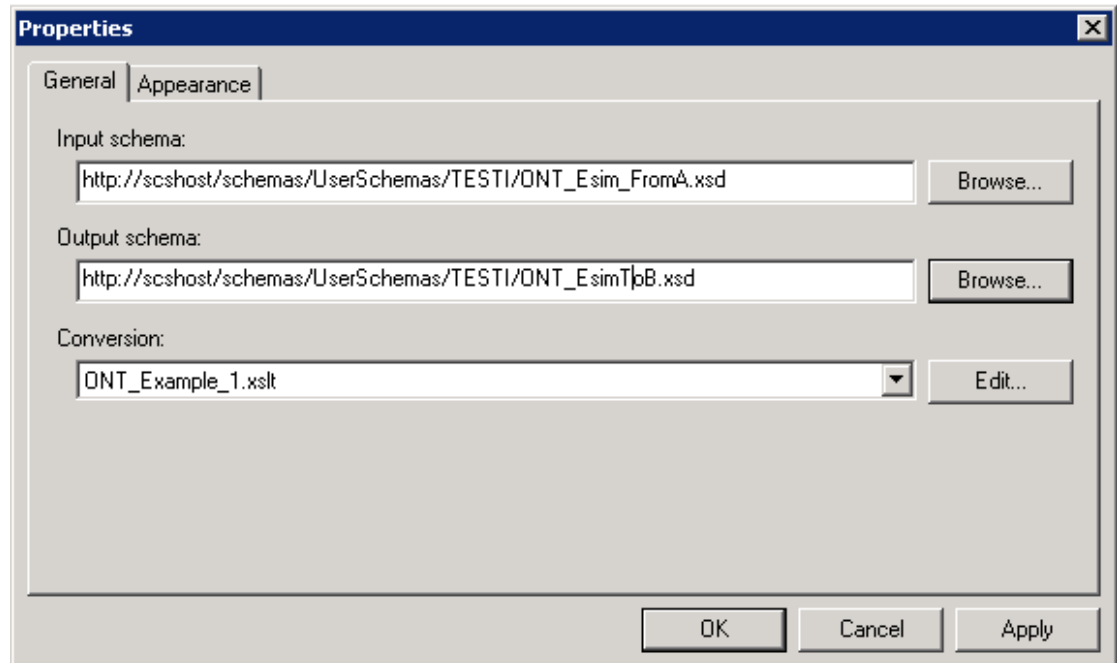
Conversion

Conversion-komponentit (yksinkertaisesti suomennettuna XML-muunnokset) on yksi keskeisimmistä workflow-designerin komponenteista. Sen avulla määritellään XML-dokumentin muunnokset (transformaatiot). Yleensä eri komponenttien väliin sijoiteltavat XML-muunnokset tulkitsevat saapuvien ja lähtevien XML-dokumenttien rakenteen XSD-tiedostojen perusteella. Niihin voi kuitenkin määritellä erikseen XSD-tiedoston, jonka perusteella XML-dokumentin rakennetta voidaan muokata. Conversion-komponenttia kuvaava symboli muodostuu kahdesta dokumenttia esittävästä kuvasta, jotka on yhdistetty viivalla (kuva 22).



KUVA 23. Conversion- eli muunnoskomponentit Workdflow Designerissa

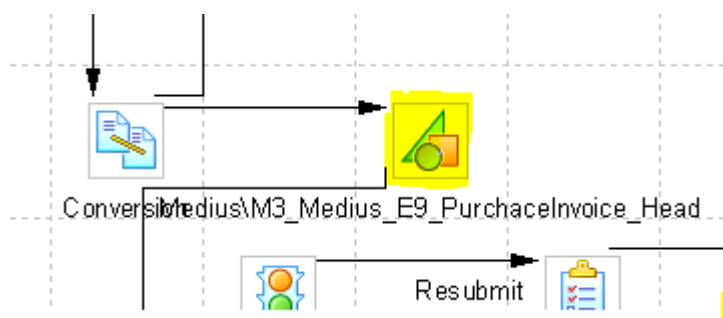
Conversion-komponentin ominaisuuksista löytyvällä Edit-painikkeella (kuva 23) voi avata Data Mapping Toolin, jossa XML-dokumentin muuntaminen toiseksi tapahtuu helposti graafisen työkalun avulla. Muunnokset tallennetaan XSLT-tiedostoina Service Connectin hakemistoon luotavaan alahakemistoon. Graafisen



KUVA 24. Conversion-komponentin ominaisuuksien säätäminen

Aliprosessi- eli sub-workflow –kutsu

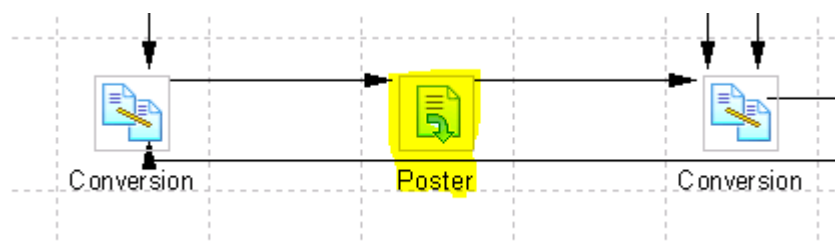
Workflow Designerissa sub-workflow –kutsulla voi sen nimen mukaisesti kutsua muita prosesseja. Tarve tälle voi syntyä esimerkiksi saapuvan XML-dokumentin sisältämien tilaustietojen rivien läpi käyminen. Sub-workflow –komponentin asetuksissa esitellään aliprosessille välittävän XML-muotoisen tiedon rakenne XML-skeematiedoston avulla. Aliprosessille välitettävä tieto osoitetaan XPath-tekniikan avulla polkumaisena viittauksena.



KUVA 25. Sub-workflown eli aliprosessin kutsu Workflow Designerissa

Poster

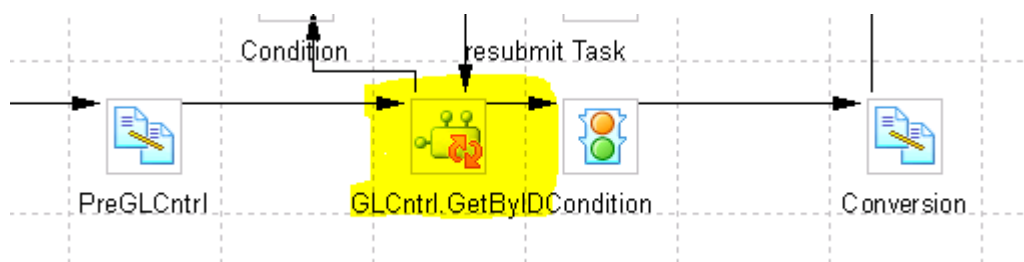
Poster-komponentti on tarkoitettu viestien lähettämiseen Service Connect Administration Consolessa määriteltyjen kanavien kautta. Poster komponentissa määritellään XML schema, jonka perusteella lähettävän XML-tiedoston rakenne muotoillaan sitä edeltävässä Conversion-elementissä. Posterkomponentin symbolina toimii Workflow Designerissa dokumentin kuva, josta lähtee vihreä nuoli vasemmalta oikealle ja kääntyy lopuksi osoittamaan alaspäin (kuva 26).



KUVA 26. Poster-komponentti Workflow Designerissa

.NET Call

.NET Call –komponentilla voidaan etäkutsua ja -suorittaa eri luokkien metodeja. Omasta mielestäni .NET Call –komponentti Conversion-komponentin lisäksi toinen erittäin olennainen osa workflow-prosesseja. Tämän komponentin avulla voidaan suorittaa Epicor 9:n eri liiketoimintaolioiden metodeja kutsumalla niitä workflow-prosessien aikana. Esimerkiksi ulkoisista järjestelmistä tulevien tietojen tallentamiseksi vaaditaan eri luokkien update- tai masterupdate –metodien kutsumista, jolloin ne suoritetaan. Parametrit välitetään metodeille osoittamalla ne XSL-muunnoksella.

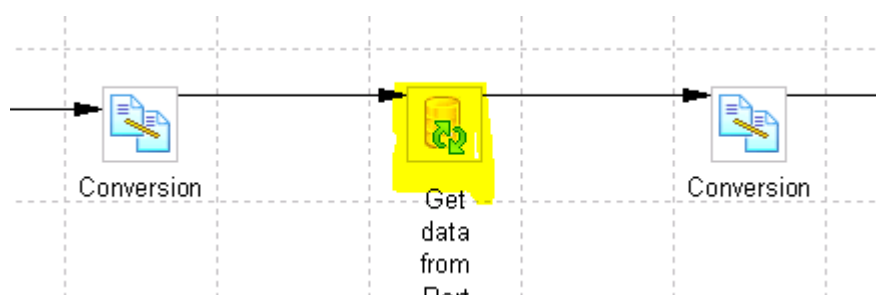


KUVA 27. .NET-kutsu Workflow Designerissa.

Kutsuttava metodi määritellään komponentin asetuksista. Service Connect Administration Consolen avulla hallinoidaan oliokirjastoja ja ne tuodaan Service connectin käytettäväksi tarvittaessa. .NET-kutsun tulee olla lisensoitu, jotta se toimisi. Muuten .NET-kutsu jää suorittamatta ja tästä syntyy virheilmoitus tapahtumalokiin.

DB Operation

Kuten jo komponentin nimestä voi päätellä, sen avulla voidaan suorittaa erilaisia tietokantakomentoja. DB operationiin voidaan kirjoittaa suoria tietokantakomentoja ja -hakuja. DB operationia ei suositella muuhun kuin tietokantakyselyihin SQL:n SELECT-komennolla. DB operation komponenttia en käyttänyt kuin kerran, kun rajapinnassa jouduttiin hakemaan ja käsittelemään hyvin suuri määrä tietoa. Tällöin syntyi tarve eritellä suoran tietokantakyselyn avulla valittavat tietokantataulun kentät.

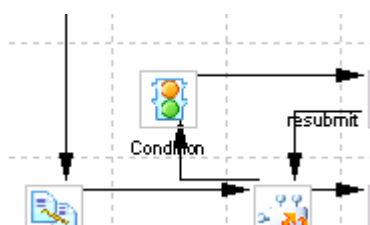


KUVA 28. DB operation

Condition

Condition-komponentin määritellään eri ehtoja, joiden täytyessä tehdään jokin tietty toiminto. Tämän komponentin avulla voidaan automatisoida toimintoja, jotka

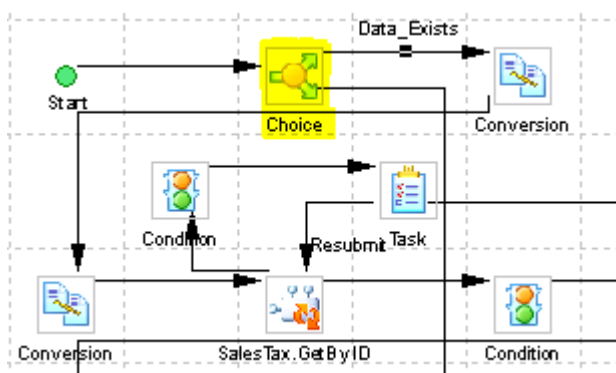
prosessin aikana suoritetaan, mikäli prosessin aikana syntyy esimerkiksi jokin virhe. Tällainen virhe voi olla esimerkiksi jonkin liiketoimintaolion suorituksen epäonnistuminen, jolloin sitä yritetään suorittaa uudelleen esimerkiksi maksimissaan 10 kertaa. Jos liiketoimintaprosessin suoritus ei onnistu kymmenenkään kerran jälkeen, prosessi jatketaan loppuun suorittamatta loppuja toimintoja.



KUVA 29. Condition-komponentti Workflow Designerissa

Choice

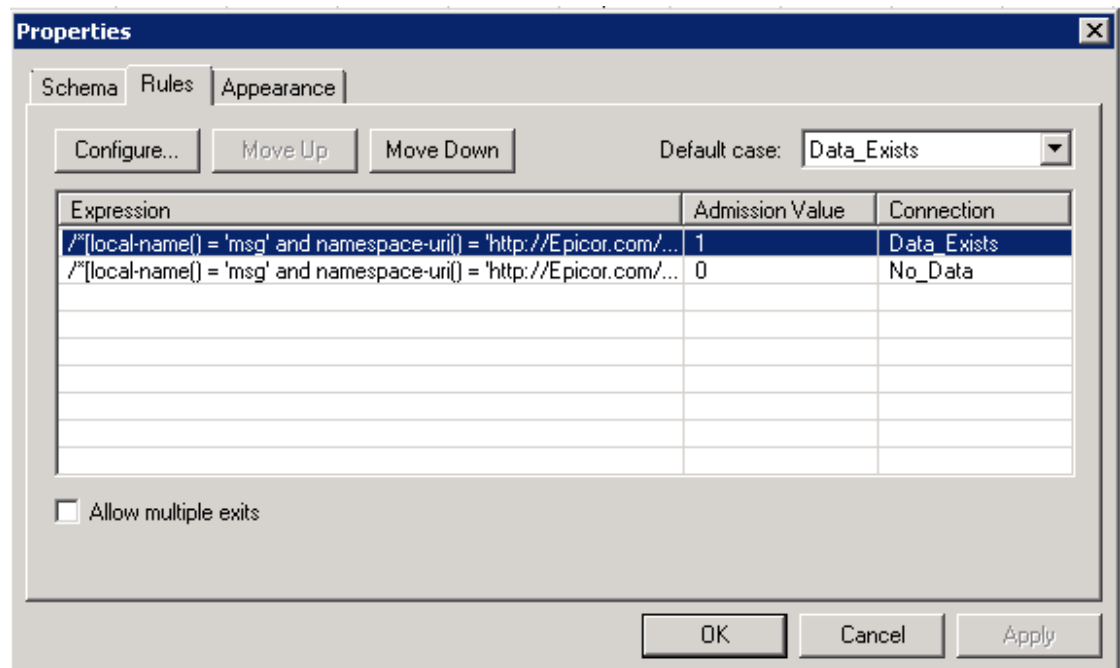
Choice-komponentin avulla voidaan luoda haaroja work-flowprosessiin siinä liikkuvan tiedon perusteella. Coice-komponentin avulla voidaan tutkia esimerkiksi sitä, onko prosessissa käsiteltävänä olevassa XML-dokumentissa kaikki tarpeelliset tiedot elementtien sisällä. Kuvassa 30 heti start-elementin jälkeen prosessissa olevasta tiedosta tarkastetaan, onko tietoa olemassa vai ei.



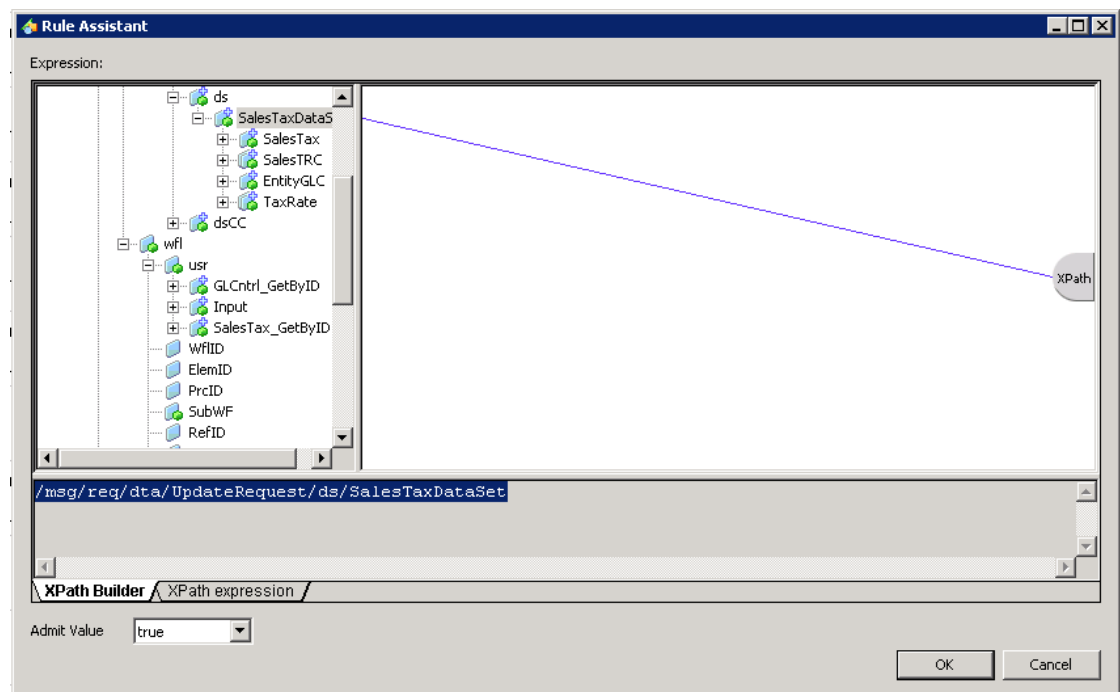
KUVA 30. Choice-elementtiä käytetään eri asioiden tutkimiseen

Choice-elementin asetuksissa voidaan määritellä säännöt, joiden perusteella, kuinka prosessia jatketaan. Nämä säännöt määritellään choise-elementin ominaisuuksissa rules-välilehdellä (kuva 31). Tuplaklikkaamalla kuvassa 31 korostettuna olevaa riviä aukeaa rule assistant –ohjelma, joka on hyvin samankaltainen Data Mapping Tool, jonka XPath-tekniikan voidaan osoittaa se elementti XML-dokumentista, jossa olevan

tiedon olemassaolo halutaan tarkistaa vetämällä viiva vasemmasta reunasta ja halutusta elementistä kuvan oikeaan reunaan jolloin ohjelma muodostaa XPath-ilmaisun tai -koodin automaattisesti.



KUVA 31. Choice-komponentin asetusten määrittely



KUVA 32. XPath-polkuviihtaukset toteutetaan graafisen työkalun avulla

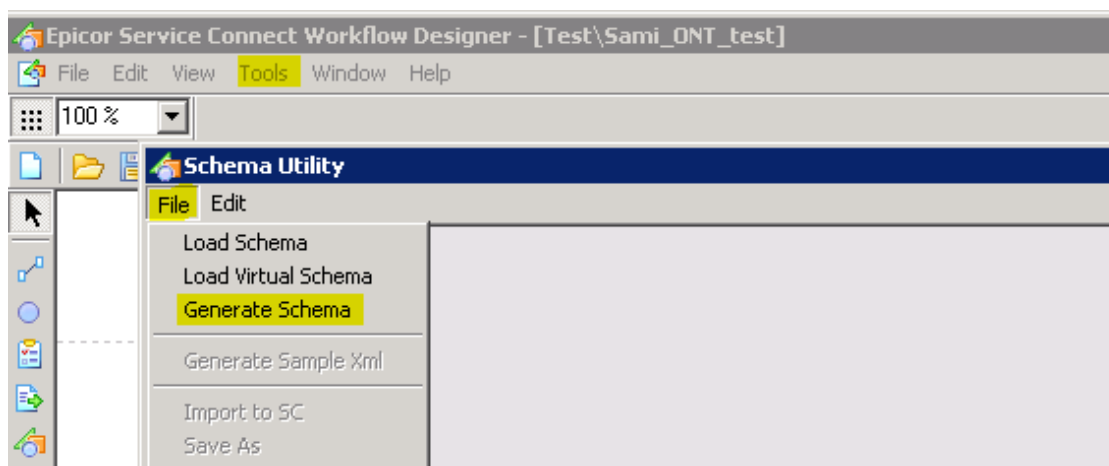
6.1.3 Schema Utility

Service Connectiin kuuluu Schema Utility –työkalu, jonka avulla XSD-tiedostojen luominen on helppoa. Schema Utilityssa työkalulle esitetään näyte XML-tiedostosta työkalun hakutoiminnolla. Näytteeksi annettava tiedosto voi olla myös esimerkiksi CSV-tiedosto, jolloin käytetään Schema Utilityn csv2xml-ohjelmaa, joka muuntaa näytteen nimensä mukaisesti CSV-muotoisesta XML-muotoon. Tässä esimerkissä luodaan XML schema yksinkertaiselle XML-dokumentille (kuva 33).

```
<tilaukset>
  <tilaus>
    <tilausrivi>
      <tilausnro>01</tilausnro>
      <asiakas>Asiakas1</asiakas>
      <tilausrivinro>1</tilausrivinro>
      <tuote>tuote</tuote>
      <tilattuKPL>3</tilattuKPL>
      <toimitettuKPL>2</toimitettuKPL>
    </tilausrivi>
    <tilausrivi>
      <tilausnro>01</tilausnro>
      <asiakas>Asiakas1</asiakas>
      <tilausrivinro>2</tilausrivinro>
      <tuote>tuote</tuote>
      <tilattuKPL>25</tilattuKPL>
      <toimitettuKPL>25</toimitettuKPL>
    </tilausrivi>
  </tilaus>
</tilaukset>
```

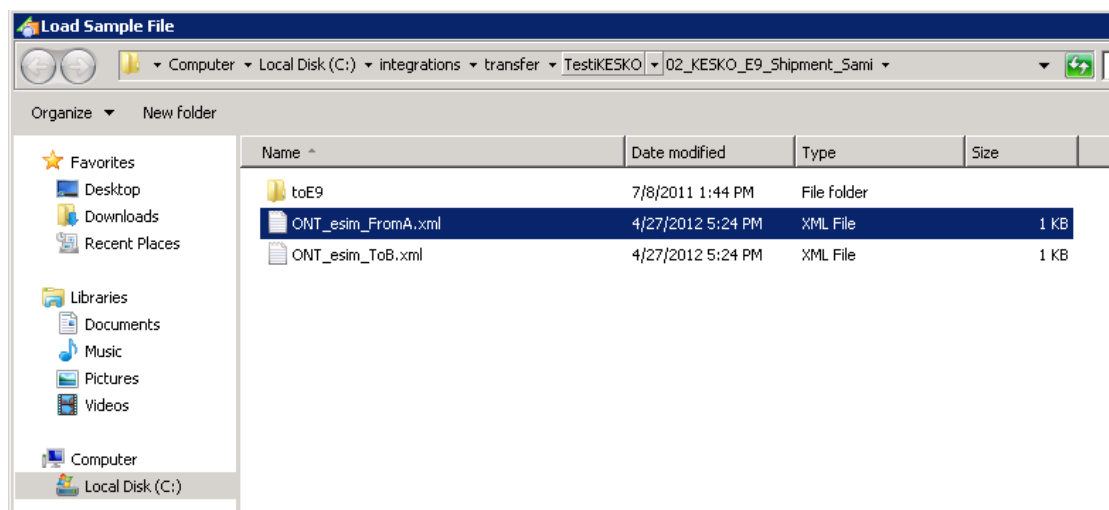
KUVA 33. Yksinkertainen XML-dokumentti

Schema Utility –työkalu löytyy Workflow Designerin työkalupalkin tools-alasvetovalikosta. Uuden schema-tiedoston generointi aloitetaan Schema Utilityn file-valikon Generate Schema –toiminnolla (kuva 34).



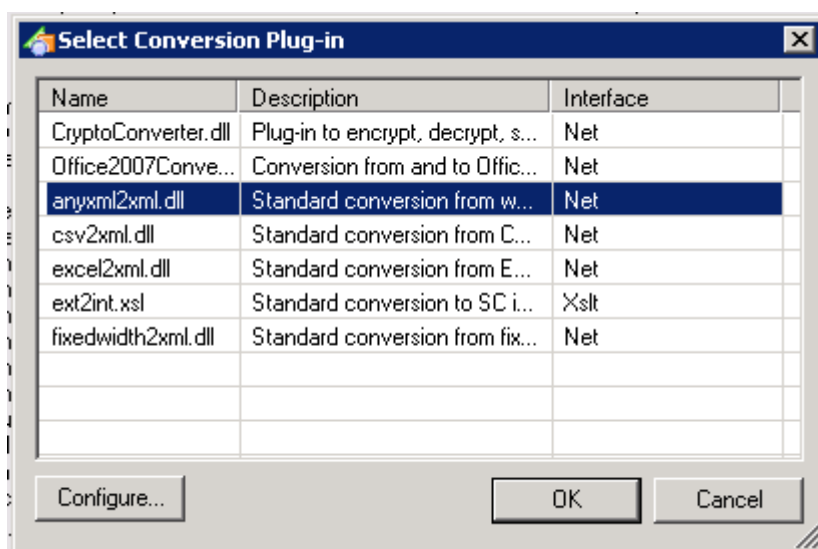
KUVA 34. Schema Utility –työkalun käynnistäminen

Seuraavaksi valitaan näytetiedosto, jonka perusteella uusi XSD-tiedosto muodostetaan. Näytetiedoston sijainti riippuu siitä, minne se on tallennettu käyttäjän toimesta tai mihin se on lähetetty toisen järjestelmän toimesta. Koska tässä tapauksessa luodaan XML schema-tiedostoa samasta tiedostosta, joka on esimerkkinä esitetty kappaleessa 5.2, valitaan se näytteeksi (kuva 35).



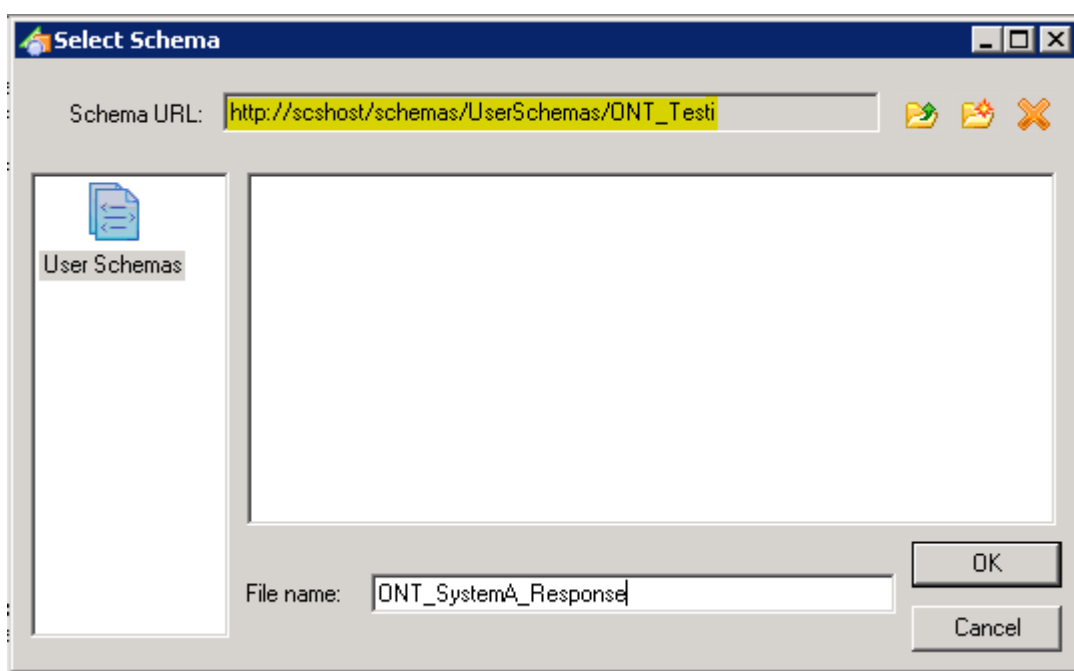
KUVA 35. Näytetiedoston hakeminen Schema Utility-toiminnossa

Kun näytetiedosto on valittu, Schema Utilityssa valitaan plug-in, jonka avulla määritellään, millainen schemasta rakennetaan. Yleisimmät vaihtoehdot ovat anyxml2xml- ja ext2int-muunnokset. anyxml2xml-muunnos –pluginin avulla luodaan schema-tiedosto, jolla kuvaillaan vapaamuotoisen XML:n rakenne. Tässä tapauksessa valitaan tämä plug-in (kuva 36).



KUVA 36. Muunnoslisäosat (conversion plug-in) Schema Utilityssa

Kun plug-in on valittu, luo Schema Utility uuden scheman XML schema-standardin mukaisesti käyttäjän valitseman XML-dokumentin perusteella. Syntyneessä XSD-dokumentissa kuvataan XML-tiedoston rakenne sekä elementtien nimet, niiden attribuutit ja elementtien sisällä välitettävien tietojen tietotyypit. Tämän jälkeen luotu schema tallennetaan Service Connectin hakemistoon (kuva 37). Nimi on käyttäjän vapaasti määriteltävissä, mutta suotavaa on, että nimi kuvaa mahdollisimman hyvin sitä, mitä tietoa XML-tiedostot, jota schema kuvaa, pitävät sisällään.



KUVA 37. Schema Utilityn muodostaman XML scheman tallennus

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://Epicor.com/SC/UserSchema"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://Epicor.com/SC/UserSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="tilaus">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="tilausrivi">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="tilausnro" type="xs:unsignedByte" />
              <xs:element name="asiakas" type="xs:string" />
              <xs:element name="tilausrivinro" type="xs:unsignedByte" />
              <xs:element name="tuote" type="xs:string" />
              <xs:element name="tilattuKPL" type="xs:unsignedByte" />
              <xs:element name="toimitettuKPL" type="xs:unsignedByte" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

KUVA 38. Yksinkertaisen XML-dokumentin schema

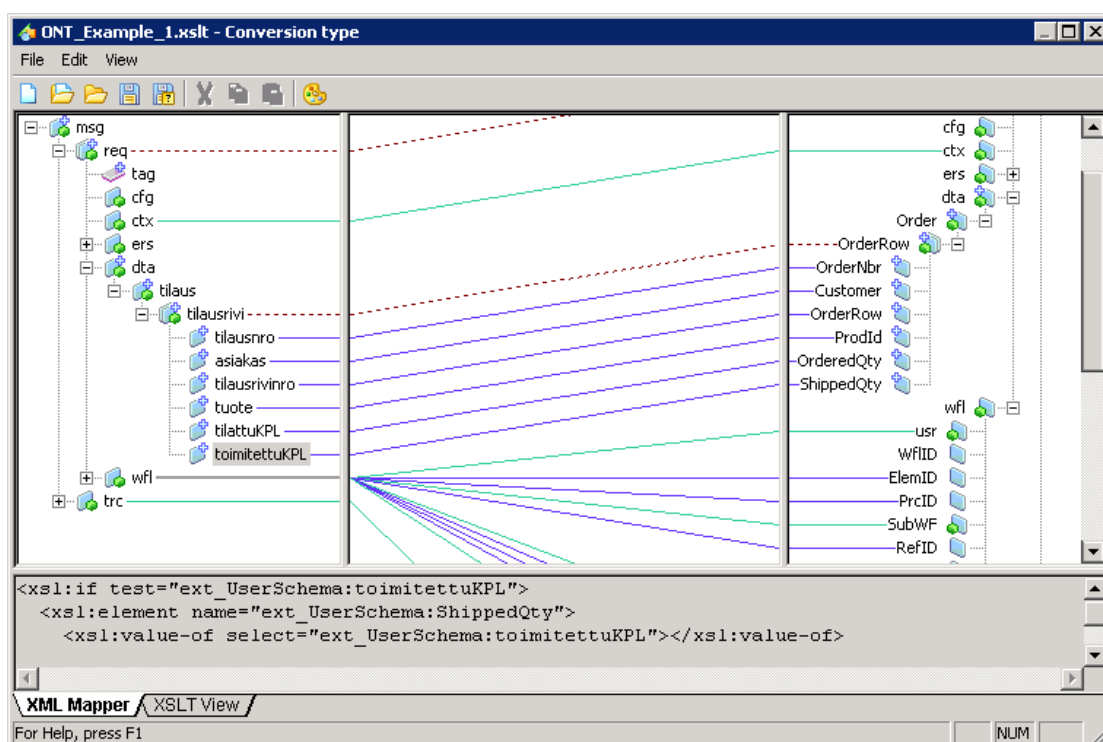
6.1.4 XML Mapper

Service Connect sisältää myös graafisen XML Mapper -työkalun, jonka avulla XML-dokumenttien muuntaminen tapahtuu. Sen tarkoitus on helpottaa rajapintojen toteuttamista ja sen avulla esimerkiksi muut järjestelmät, jotka tukevat XML-standardia, on helposti integroitavissa E9:ään. Service Connect tosin tarjoaa mahdollisuuden muuntaa muussa muodossa vastaanotettavat tiedot XML-muotoon. XML mapper hyödyntää XML-standardin mukaisia tekniikoita (XML, XSLT, XPath ja XML schema (myös XSD)). Työkalulla muunnetaan XSL-muunnosten avulla XML-dokumentit Conversion-komponentin asetuksissa määriteltävien schemojen mukaan uudelleen. Tämä auttaa järjestelmien välillä liikkuvan tiedon muokkaamista.

XML Mapperin käyttöliittymä on hyvin yksinkertainen. Käyttöliittymän molemmissa reunoissa on esitetty puurakenteena prosessissa kulkeva tieto. Käyttäjä valitsee vasemmasta reunasta tiedon haluamastaan elementistä klikkaamalla ja raahamalla näin syntyvän viivan oikeaan reunaan haluaamansa elementtiin. Näin elementtien välille syntyy viiva. Viiva kuvaa tiedon kulkua elementtien välillä. Käyttäjän mappauksen

perusteella ohjelma luo samalla XSLT-merkintäkieltä. Ohjelman synnyttämä XSLT on sen 1.0 version mukaista.

Kuva 39 on ruudunkaappaus XML-mapperin ikkunasta, josta hyvin näkyy sen käyttöliittymä. Tässä tapauksessa XML Mapperissa on toteutettu XSL-muunnos, jonka avulla kappaleessa 5.2 esimerkkinä yksinkertaisesta XML dokumentista saadaan aikaiseksi uusi XML-dokumentti. Lähdetiedon rakenne tulkitaan siis aiemmin luodun XML scheman perusteella ja kohdetiedon rakenne on kuvattu toisella samalla tavalla luodulla schemalla.

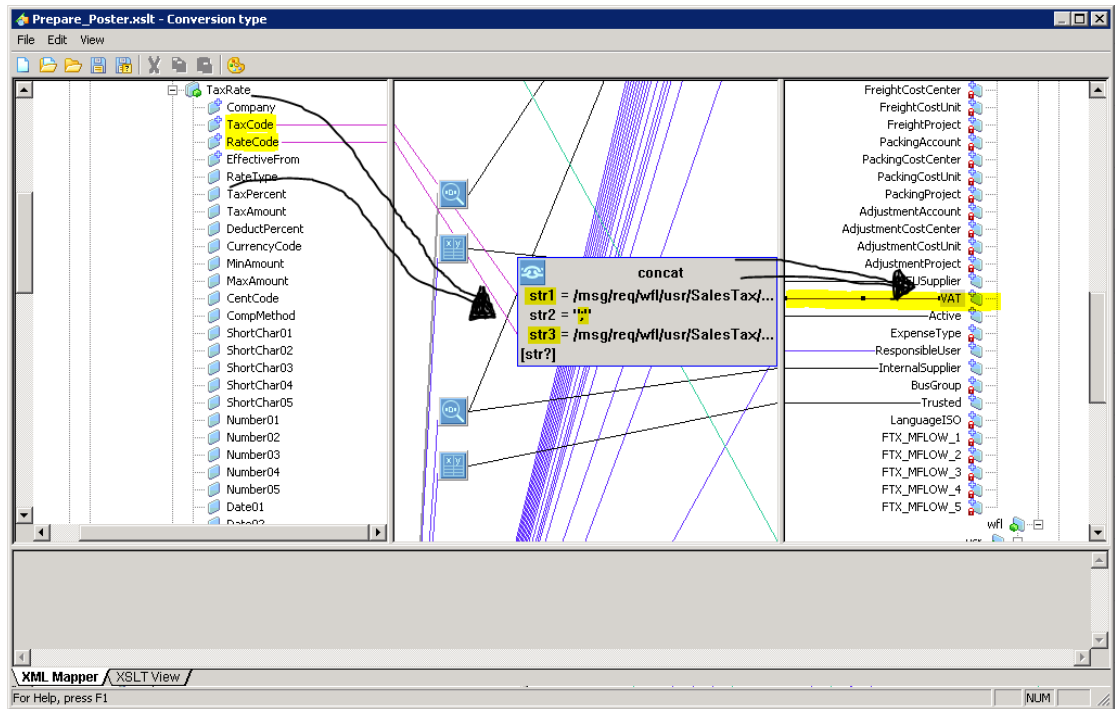


KUVA 39. XML Mapper

Eri väriset viivat edustavat erilaisia XSLT-kuvauskielen eri ilmauksia. Esimerkiksi yhtenäinen sininen viiva kuvaa XSLT:n value-of elementtiä, eli elementin arvon valitsemista uuden luotavan elementin sisällä kuljetettavaksi tiedoksi. Katkoviiva kuvassa 36 taas on XSLT:n for-each –elementti, jonka avulla ilmaistaan, että jokaisen tilausrivin lapsielementit tiedot valitaan muunnoksessa.

Functoidit XML Mapperissa

Functoidit ovat XML Mapperissa muunnoksen väliin sijoitettavia pieniä ohjelmia, joiden avulla voidaan suorittaa erilaisia XML-tekniikoihin perustuvia funktioita. Niiden avulla voidaan esimerkiksi toteuttaa merkkijonojen yhdistäminen.



KUVA 40. Functoidien avulla voidaan suorittaa XML-tekniikkaan kuuluvia funktioita

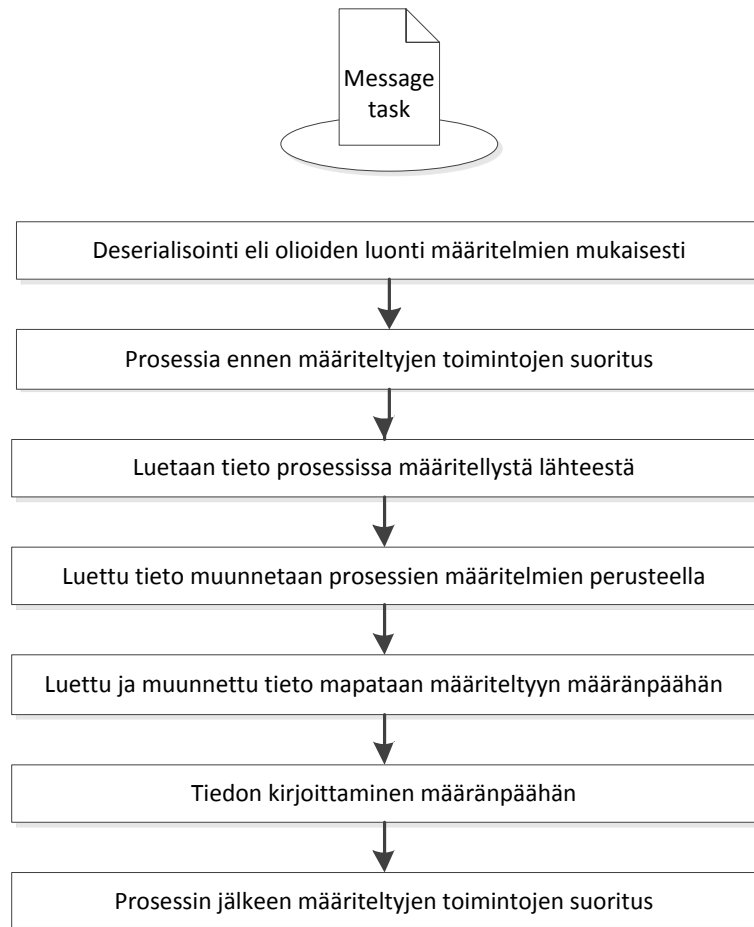
Merkkijonojen yhdistäminen XML Mapperissa toteutetaan concat-functoidin avulla (kuva 40). Functoidiin valitaan yhdistettävät merkkijonot vetämällä viiva functoidin muuttujiin vasemmasta reunasta, kuten kuvan esittämässä tapauksessa on tehty) tai kovakoodaamalla, jolloin muuttajiin määritellään staattinen arvo (kuvassa `str2 = '';`). Näin ollen VAT-nimisen `xsl:element`in arvoksi tulee esimerkiksi `"taxcode;ratecode"`, joista merkkijonot `"taxcode"` ja `"ratecode"` ovat prosessissa liikkuvan tiedon mukaan muuttuvia arvoja. XML Mapper muodostaa kaiken aikaa taustalla XSLT:n version 1.0 mukaista merkkäuskieltä.

6.2 Medius Integration Gateway

Medius Integration Gateway (MIG) on Mediuksen toteuttama middleware, joka on kehitetty helpottamaan MediusFlow:n integrointia ulkoisiin järjestelmiin. Se tarjoaa joukon erilaisia adaptoreita ja toimintoja, joiden avulla MediusFlow:n integrointi ulkoisiin järjestelmiin on mahdollista toteuttaa.

Medius Integration Gateway on Epicor Service Connectiin verrattuna huomattavasti kevyempi middleware-ratkaisu, joka tarjoaa vain ytimen erilaisten toimintojen suorittamiseen. Varsinaiset prosessit on määritelty erillisten XML-dokumenttien avulla. Niiden perusteella on määritelty prosessit ja se, mistä tietoa haetaan, miten muunnetaan ja minne se lähetetään.

Lyhyesti kuvailtuna Medius Integration Gateway lukee alussa prosessin, deserialisoi sen eli luo siitä oliot, jonka jälkeen prosessi suoritetaan sovelluksessa (kuva 41).



KUVA 41. Yksinkertaistettu kuva prosessien suorittamisesta Medius Integration Gatewayssa (Medius 2012)

Prosessien määrittely

Medius Integration Gatewayssa suoritettavat prosessit määritellään XML-muodossa., jonka jälkeen asetukset tallennetaan XML-dokumenttina haluttuun kansioon. Tallennuskansio sijaitsee ennalta määritellyssä hakemistossa. Medius Integration Gatewayssa on kaksi erilaista prosessityyppiä: transformation-ja ja message-tyyppi. E9-Medius-integraatiossa on käytetty message-tyyppisiä prosesseja.

Message-tyypin prosessi kuvaillaan message-elementin lapsielementeissä Message-tyypin prosessiin määriteltävät asetukset on käyty läpi tarkemmin Mediuksen omissa dokumenteissa.

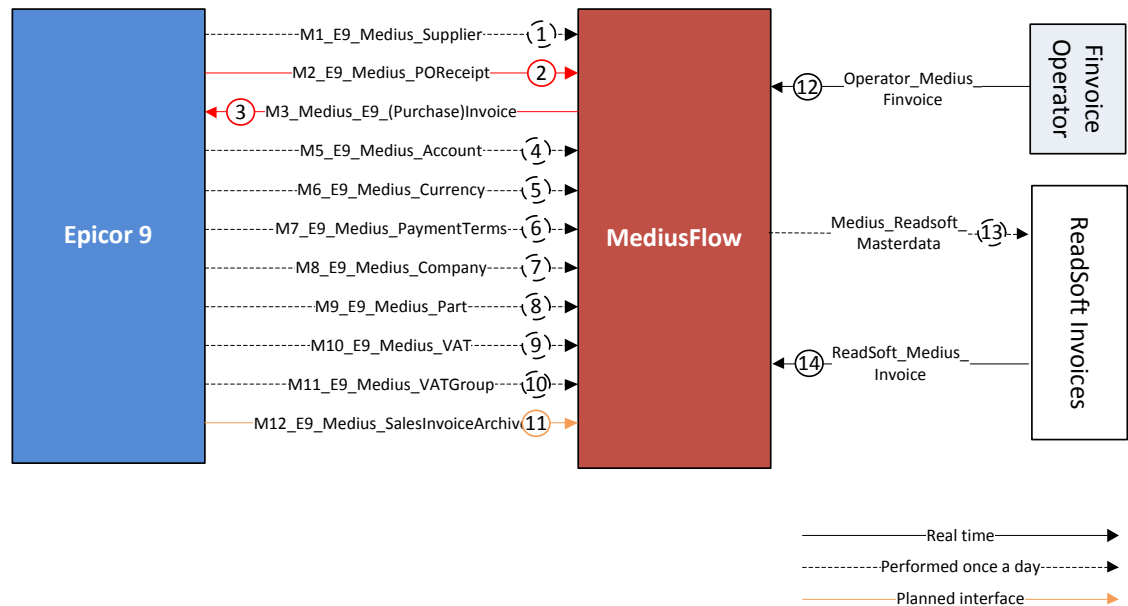
Schedule

Medius Integration Gateway suorittaa edellä määrätyt prosessit XML-tiedostoon tallennettavien määritysten mukaisesti. Medius Integration Gatewayn asennuksen yhteydessä perustetaan Windows-palvelu komentorivillä. Komennolle, jonka avulla Medius Integration Gateway Service perustetaan, annetaan joukko parametreja, joista yksi on tiedostopolku perustettavan palvelun aikana suoritettavien prosessien kuvaus ja aikataulutus.

7 EPICOR 9–MEDIUSFLOW-INTEGRAATIO

Kuvatus integraation keskeisimmät järjestelmät ovat Epicor 9 (jatkossa myös lyhennetty E9) ja MediusFlow. Näiden järjestelmien lisäksi kokonaisuuteen kuuluu ruotsalaisen ReadSoftin tekstintunnistusovellus (OCR eli Optical Character Recognition) ReadSoft Invoices, jonka integroinnista MediusFlow:hun on vastannut Medius itse aikaisemmissa projekteissaan. Lisäksi vaadittiin myös rajapinnan luomista finvoicen vastaanottamista varten. Kuvan 42 yleiskuvauksessa finvoicen lähettäjä on abstraktisti nimetty operaattoriksi. Finvoice on Suomessa käytössä oleva sähköisen laskun standardi. Operaattorille en ole määritellyt tässä tapauksessa nimeä, sillä se riippuu siitä, mikä yritys finvoicen lähettää.

Tarve integraatiolle syntyi, kun Smart Time Oy alkoi tarjota MediusFlow:ta ostolaskuprosessin sähköiseksi ratkaisuksi yrityksille, jotka olivat valinneet Epicor 9:n ERP-järjestelmäksi. MediusFlow on integroitu aiemminkin eri järjestelmiin (mm. Epicorin iScala) aiemminkin eri projekteissa. Integraatoratkaisua Epicor 9:n ja MediusFlow:n välillä ei kuitenkaan ollut olemassa, joten integraatio päätettiin toteuttaa Smart Timen toimesta.



KUVA 42. Yleiskuvaus integraatiosta ja rajapinnoista

Rajapintojen nimeämiskäytäntö

Smart Timella rajapintojen nimeämiseksi Epicor-projekteissa on ollut aina tietty käytäntö. Tietyn nimeämiskäytännön tarkoituksena on luoda yhtenäinen ja helposti tunnistettavissa oleva tapa nimetä rajapinnat siten, että pelkän nimen perusteella voidaan nähdä se, mistä tieto tulee, minne se menee ja mitä rajapinnan kautta kulkeva tieto pitää sisällään. Esimerkiksi rajapinnan M1_E9_Medius_Supplier-nimestä voidaan nähdä, että rajapinnan kautta kulkee toimittajan perustiedot E9:stä MediusFlow:hun. M-kirjain rajapinnan numeron edessä päätettiin lisätä, jotta rajapinta on selkeästi tulkittavissa MediusFlow:n ja Epicor 9:n välille toteutetun integraation rajapinnaksi.

7.1 Perustietojen ylläpito

Laskuprosessien siirtyessä paperisesta sähköiseen muotoon korostuu järjestelmissä olevien perustietojen merkitys. Riski erilaisille käsittelyvirheille vähenee, mitä paremmin prosessi on suunniteltu, järjestelmä parametroitu ja ohjaustiedot ylläpidetty. (Lahti & Salminen 2008)

Haasteellisimpia asioita perustietojen integroinnissa järjestelmien välillä on tietojen parametointi niin, ettei mitään tietoa tarvitsi niin sanotusti kovakoodata. Perustietojen ollessa mahdollisimman parametroitua rajapinnat toimivat parhaimmillaan sellaisenaan missä tahansa ympäristössä, johon on asennettu nämä järjestelmät.

Perustietojentietojen ylläpito integraatiossa tapahtuu E9:ssä, josta ne päivitetään MediusFlow:hun. MediusFlow:n välitettävät pakolliset perustiedot ovat:

- Yrityksen perustiedot (Company)
- Tilikartta tai pääkirjanpidon tilit (Accounting)
- Valuutat ja niiden vaihtokurssit (Currency ja exchange rates)
- Maksuehdot (Payment terms)
- Arvonlisäverotukseen liittyvät tiedot ja ALV % (VAT)
- Toimittajan perustiedot (Supplier tai Vendor)
- Tuotteiden perustiedot (Part)
- Ostotilauksen otsikko, rivit ja tiliöintitiedot

Perustietojen ylläpito päädyttiin toteuttamaan siten, että tiedot E9:stä MediusFlow:hun välitetään XML-dokumentteihin tallennettuna. Perusajatuksena oli luoda Service Connectiin sisältyvän Workflow Designerin avulla prosessi (workflow) jokaista MediusFlow:n vaatimaa perustietoa kohden. Nämä prosessit sidottiin E9:n liiketoimintaolioiden Update-metodeihin. Taulukossa 1 on lueteltu ne liiketoimintaolioiden Update-metodit, joihin on sidottu perustiedot ulos E9:stä lopulta välittävät workflow:t. Rajapintojen nimeämisessä numerointi määräytynyt toteuttamisjärjestyksen mukaan.

TAULUKKO 1. Metodeihin linkitettyt rajapinnat

Perustieto	Metodi	Workflow
Toimittajan tiedot	Vendor.Update	M1_E9_Medius_Supplier
Ostotilausten kuittaukset	Receipt.Update	M2_E9_Medius_POReceipt
Ostotilaukset	PO.Update	M4_E9_Medius_PO
Tilikirjan dimensiot	COASegValues.Update	M5_E9_Medius _Account ja _AccountLines
Valuutat ja kurssit	Currency.Update	M6_E9_Medius_Currency

Ostojen maksuehdot	PurTerms.Update	M7_E9_Medius _PaymentTerms
Yrityksen tiedot	Company.Update	M8_E9_Medius_Company
Tuotteet	Part.Update	M9_E9_Medius_Part
Arvonlisäverot	SalesTax.Update	M10_E9_Medius_VAT

XML-dokumentit lähetetään Workflow Designerissa yleensä prosessin loppuun sijoitettavan poster-komponentin avulla (esitelty aiemmin luvussa 6.1.3). Poster-komponentissa määritellään kanava, jonka osoittamaan kansioon XML-tiedosto lähetetään. Kanavien asetukset säädetään Service Connectin yhteysasetuksissa. Tätä ennen rajapintakohtaisesti on määritelty joukko toimintoja eri tarkoituksiin, kuten liiketoimintaolioiden eri metodien kutsuun.

Kun tiedot on tallennettu, lähetetään ne siis XML-tiedostona tiettyyn kansioon, josta ne haetaan myöhemmin MediusFlow:hun. Tämä toiminto on toteutettu Medius Integration Gatewayn avulla. Tätä toimintoa varten on määritelty joukko prosesseja, jotka suoritetaan ennalta määrätyn aikataulun mukaisesti. Toimintojen suorituksesta vastaa Medius Integration Gatewayn asennuksen yhteydessä luotava Windows-palvelu (service). Palvelun asetukset luetaan XML-tiedostosta, joka sijaitsee MIG:n tallennuskansiossa. Myös nämä asetukset ovat vapaasti määriteltävissä, kunhan ne on toteutettu MIG:n käyttöohjeissa kuvatulla tavalla.

Liitteessä 2 on kuvattu tämä prosessi kokonaisuudessaan Microsoft Visiolla toteutetun kaavion avulla. Vasemmassa reunassa sijaitseva sininen suorakulmio kuvaa Service Connectia ja sen avulla toteutettuja rajapintoja. Oikeassa reunassa puolestaan punaisena suorakulmiona kuvattu Medius Integration Gateway. Siniset viivat kuvaavat E9:stä MediusFlow:hun ja punaiset viivat MediusFlow:sta E9:ään liikkuvaa dataa. Oikeassa reunassa sijaitseva kuvaa tietokannan tauluja, joihin MediusFlow:hun vietävä tieto kirjoitetaan, joista se Medius Integration Gatewayn sisäisten prosessien suorituksen yhteydessä luetaan. Nämä MediusFlow:n tietokannan NAV-alkuiset taulut tarjoavat rajapinnan MediusFlow:hun tallennettavalle tiedolle, minkä johdosta niitä kutsutaan myös rajapinta- tai validointitauluiksi. Tässä integraatiossa käytetyt NAV-integraatiotaulut esitelty taulukossa 2.

TAULUKKO 2. MediusFlow:n integraatiotaulut

Taulut	Kuvaus
NAV_COMPANY	Yrityksen perustiedot.
NAV_ACCOUNTING	Tilikirjan dimensiot
NAV_CURRENCY	Valuutat ja vaihtokurssit
NAV_SUPPLIER	Toimittajan perustiedot
NAV_PURCHASEORDER_HEAD	Ostotilauksen otsikko
NAV_PURCHASEORDER_ROW	Ostotilauksien rivit
NAV_PURCHASEORDER_ACCOUNTING	Ostotilauksen tiliöintitiedot
NAV_VAT	ALV-tiedot.
NAV_PART	Tuotteiden perustiedot

7.2 Esimerkki: M1_E9_Medius_Supplier

Tässä kappaleessa esittelen toimittajien tiedon viemiseen ja ylläpitämiseen toteutetun rajapinnan ja sen, kuinka tieto käyttäjän tallentaessa kulkee E9:stä MediusFlow:hun. Käyttäjän tallentaessa tiedot E9:n käyttöliittymässä Supplier Maintenance –ohjelmalla järjestelmä suorittaa Vendor-luokan Update-metodin. Tällöin kutsutaan synkronisesti (samaan aikaisesti) M1_E9_Medius_Supplier workflow:ssa kuvattua prosessia. Workflow:n aikana suoritetaan joukko liiketoimintaolioiden metodeja tarvittavien tietojen noutamiseksi E9:n tietokannan tauluista, jonka jälkeen tiedot yhdistellään XSL-muunnoksen avulla ja lähetetään poster-komponentin avulla Mediuksen integraatiohakemistoon odottamaan vastaavan MIG-prosessin suorittamista. Liitteessä 3 tiedon kulku prosessissa on kuvattu uimaratakaaviona.

Rajapinnan tekemiseen kuuluvat seuraavat vaiheet:

- Ns. rajapintatriggerin luominen
- Outputkanavan luominen Administration Consolessa
- Workflow-prosessin M1_E9_Medius_Supplier luominen Medius-paketin alle.
- Input Scheman luominen schema utilityn avulla, ellei sitä ole jo olemassa.
- Workflow-prosessin toteutus

– Mediusprosessin luominen (MsgMedius_Supplier.xml)

Pyrkimyksenäni on kuvata rajapinnan toteutus ja siihen liittyvät vaiheet hieman tarkemmin. Tämän ei ole kuitenkaan tarkoitus olla täysiverinen tekninen dokumentti kyseisestä rajapinnasta ja joitakin tekovaiheeseen liittyviä välivaiheita on saatettu jättää kokonaan tai osittain kertomatta.

Yksi selvitettävistä asioista perustietoja välittävien rajapintojen luomisessa oli selvittää kuinka MediusFlow integroidaan ERP-järjestelmiin. Apuvälineinä tässä selvitystyössä toimivat Mediuksen omat tekniset dokumentit ja käyttöohjeet, joiden perusteella selvisi, että MediusFlow:n tietokannan NAV-alkuiset taulut on tarkoitettu tiedon tuomiseen ulkoisista järjestelmistä. Näistä tauluista NAV_SUPPLIER-taulu (liite 5) toimii rajapintana toimittajien perustietojen tuomiseksi MediusFlow:hun. XML schema, jonka perusteella E9:stä ulos tulevan XML-dokumentin rakenne on määritelty, on kirjoitettu käsin taulun rakenteen perusteella (kuva 43). Näin E9:stä tulostettavan XML-dokumentin sisältämät elementit voidaan helposti mapata MediusFlow:n prosesseissa.

```

<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Supplier" >
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CompanyId" type="xs:string" />
        <xs:element name="SupplierId" type="xs:string" />
        <xs:element name="Name" type="xs:string" />
        <xs:element name="Name2" type="xs:string" />
        <xs:element name="Address" type="xs:string" />
        <xs:element name="City" type="xs:string" />
        <xs:element name="Zip" type="xs:string" />
        <xs:element name="Country" type="xs:string" />
        <xs:element name="Telephone" type="xs:string" />
        <xs:element name="Fax" type="xs:string" />

        <xs:element name="VAT" type="xs:string" />
        <xs:element name="Active" type="xs:string" />
        <xs:element name="ExpenseType" type="xs:string" />
        <xs:element name="ResponsibleUser" type="xs:string" />
        <xs:element name="InternalSupplier" type="xs:integer" />
        <xs:element name="BusGroup" type="xs:string" />
        <xs:element name="Trusted" type="xs:string" />
        <xs:element name="LanguageISO" type="xs:string" />
        <xs:element name="FTX_MFLOW_1" type="xs:string" />
        <xs:element name="FTX_MFLOW_2" type="xs:string" />
        <xs:element name="FTX_MFLOW_3" type="xs:string" />
        <xs:element name="FTX_MFLOW_4" type="xs:string" />
        <xs:element name="FTX_MFLOW_5" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

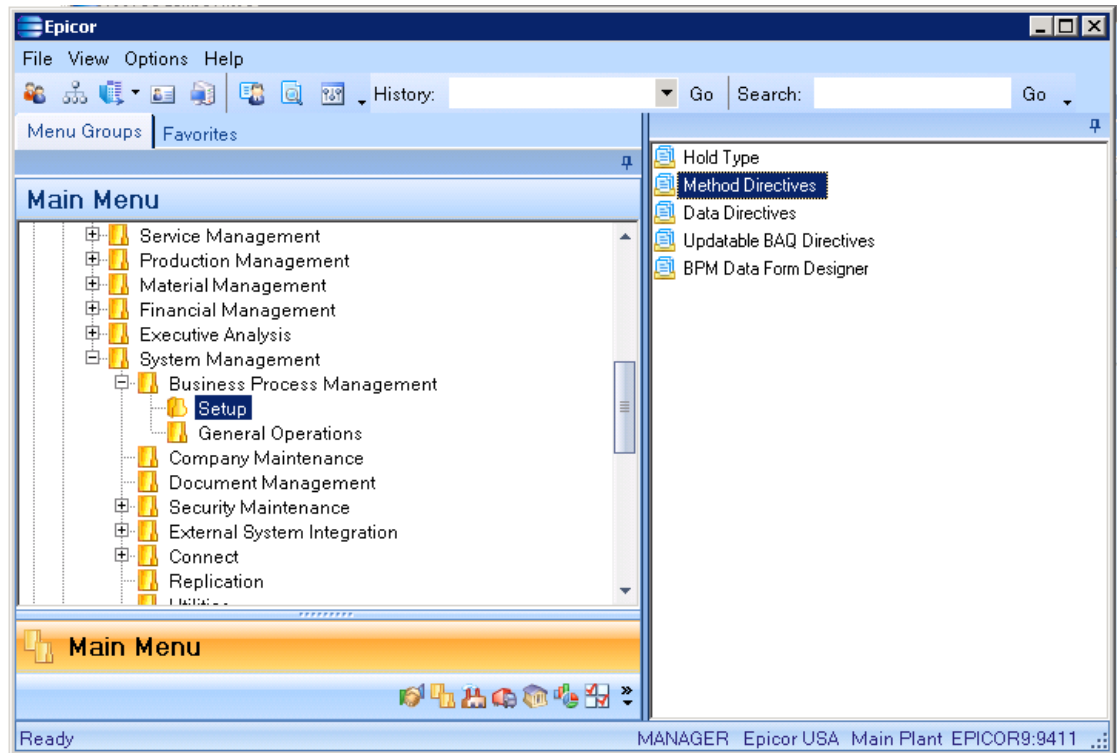
KUVA 43. Supplier.xsd

Scheman kuvaamiseen käytetyt XSD-tiedostot, kuten kuvasta 40 voidaan nähdä, ovat aika yksinkertaisia eikä niiden perusteella luotavat XML-tiedostot ole rakenteeltaan kovin monimutkaisia. Jokainen XML-dokumentti sisältää vain yhden toimittajan tiedot. Kuvassa 40 XSD-dokumentti on todellisuudessa pidempi. Katso liitteestä 4, että mitkä kentät integraatiossa on tällä hetkellä mukana myös XML schema – tiedostoissa.

7.2.1 Triggerin luominen

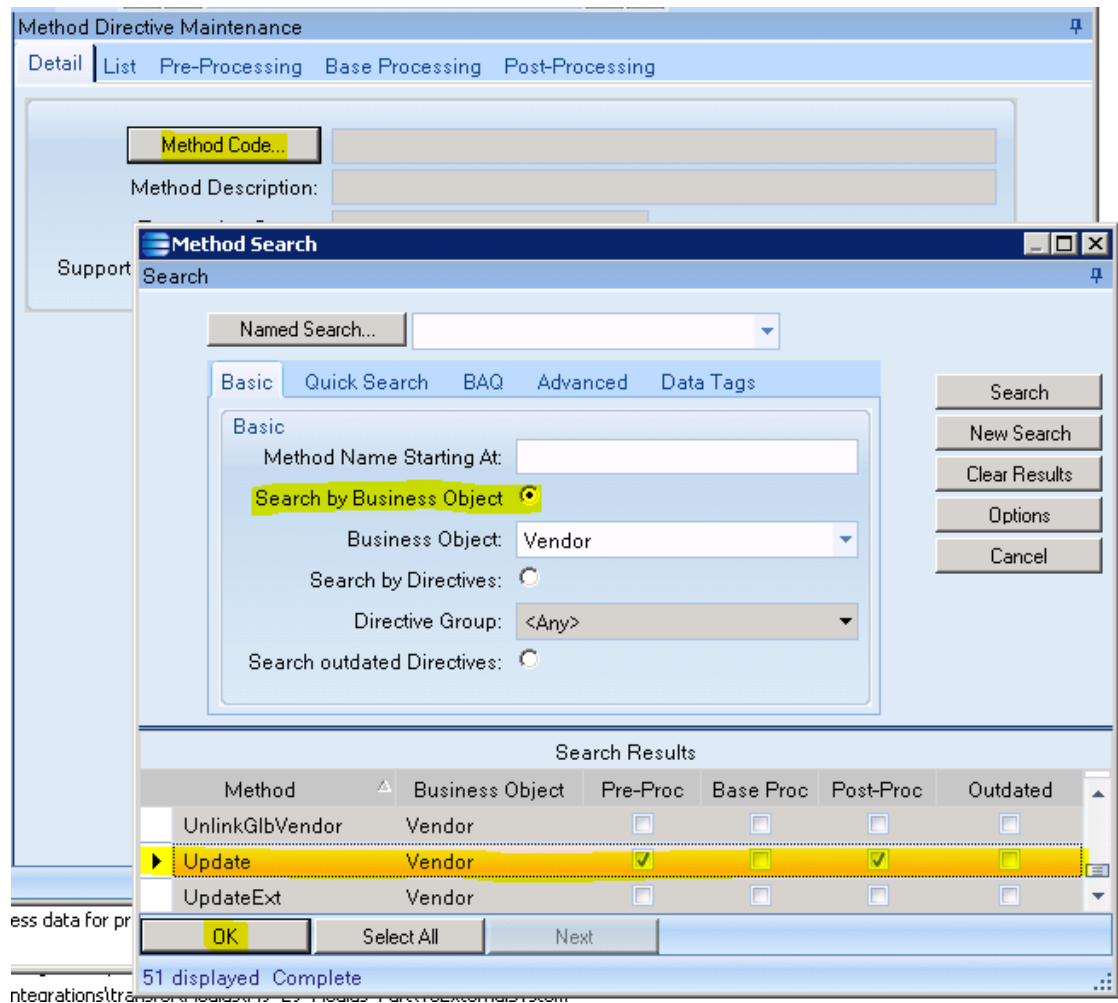
Workflow-prosessin suorittamiseksi tarvitaan aina jokin trigger eli laukaisin. Laukaisimena voi toimia joko tiedosto, joka tiettyyn hakemistoon ilmestyessään (toisen järjestelmän lähettämänä) käynnistää workflow-prosessin. E9:stä ulospäin

tallennettava tieto laukaistaan aina samalla, kun tiedon tallennuksesta vastaava asiaankuuluvan liiketoimintaolion update-metodi suoritetaan.



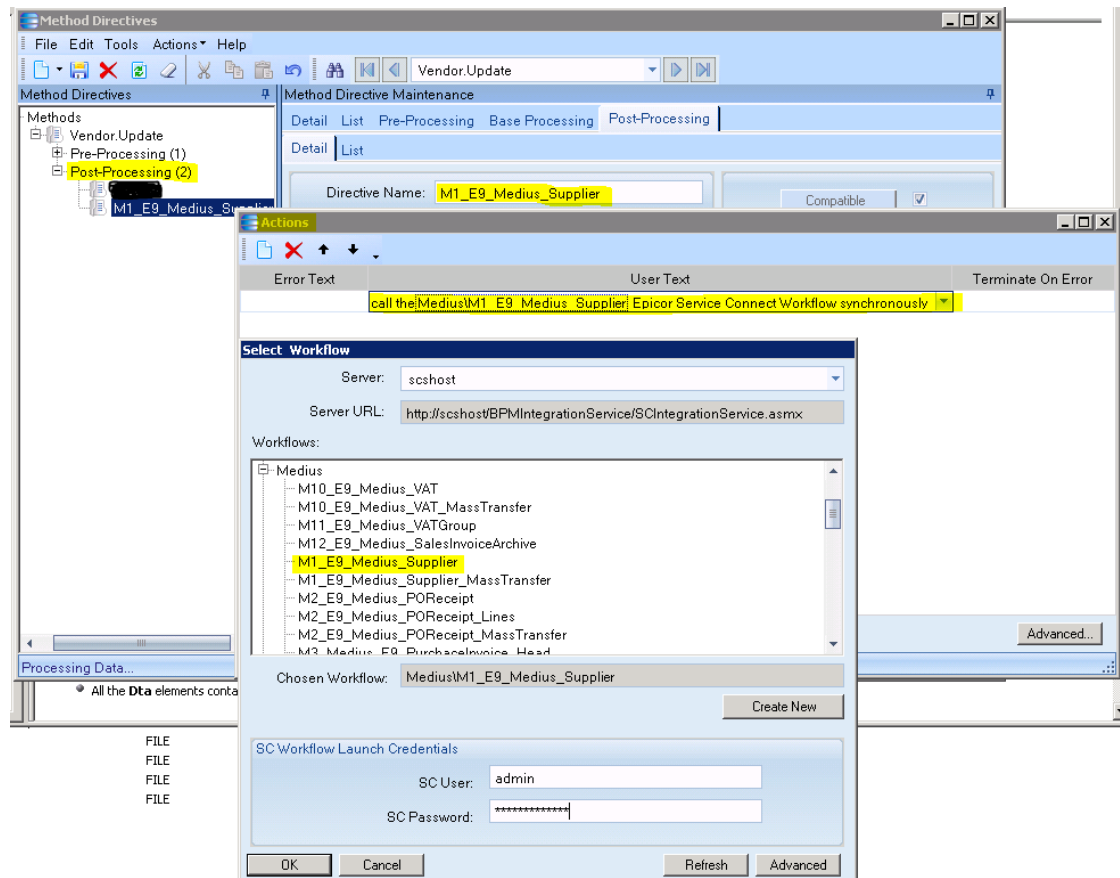
KUVA 44. Method directives

Liiketoimintaolioiden metodeille voidaan asettaa uusia toimintatapoja eli direktiivejä. Näitä hallinnoidaan Epicor 9:n käyttöliittymän kautta Method Directives-osion avulla, jonne navigointi on kuvattu kuvassa 44. Esimerkkinä esitettävä workflow-prosessi liitetään siis osaksi Vendor.Update-metodia (usein metodeista puhuttaessa käytetään ilmaisutapaa, jossa liiketoimintaolio ja siihen kuuluva metodi on erotettu pisteellä). Käyttöliittymässä haetaan ensin metodin koodi liiketoimintaolioon perustuvan haun avulla. Tämän jälkeen syntyy listaus kaikista kyseisen liiketoimintaolion metodeista. Näistä valitaan siis Update-metodi.



KUVA 45. Vendor.Update-metodi haetaan Method Search -toiminnon avulla

Vendor.Updatelle määritellään uusi prosessi (workflow), jota kutsutaan aina. Tämä tapahtuu luomalla uusi post-processing (varsinaisen prosessin jälkeen suoritettava toiminto) käyttöliittymään rakennetuilla työkaluilla, jolloin suoritettavan metodin koodiin (oletettavasti) lisätään uusia toimintoja.

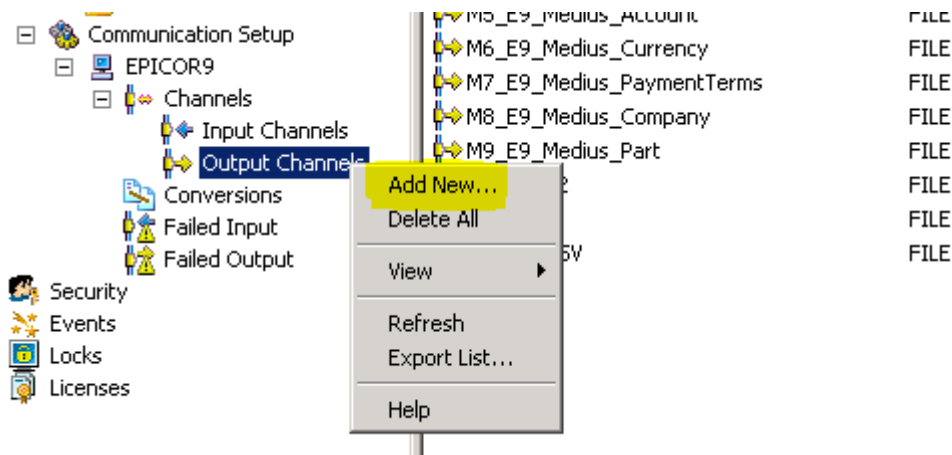


KUVA 46. Workflow-prosessin valinta

Tässä tapauksessa workflow-prosessin suorittamiseksi ei olla luotu mitään ehtoja, joten se suoritetaan aina samalla, kun Vendor.Update-metodi suoritetaan käyttäjän tallentaessa uutta tai jo olemassa olevaa tietoa. Metodidirektiiveissä voidaan myös määritellä pre-processing-toimintoja tarkistamaan esimerkiksi sitä, lisätäänkö tietokannan vendor-tauluun uusi rivi tai muokataanko olemassa olevaa riviä.

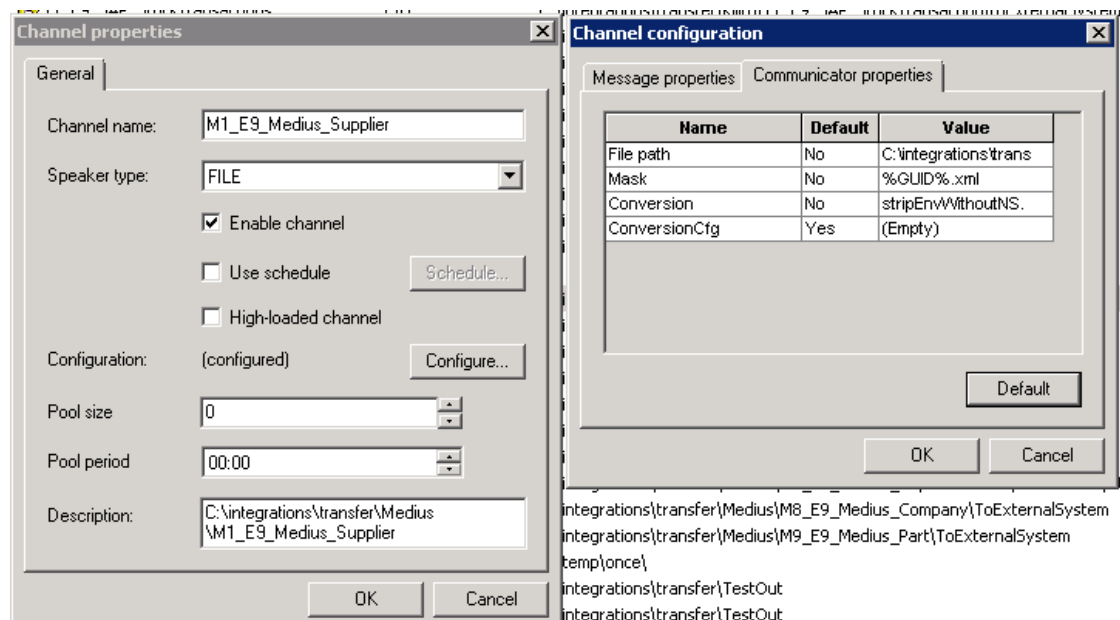
7.2.2 Output-kanavan luominen

Output-kanavalla tarkoitetaan kanavaa, jonka kautta tiedot lähetetään Epicorin järjestelmästä ulos. Koska lähettäjänä on järjestelmä itse, ei kommunikaatioasetuksissa tarvitse luoda erillistä lähettäjää. Kanavan luominen lähtee liikkeelle navigoimalla Service Connect Administration Consolessa communication-setupin output chaneleihin. Toista hiiren painiketta klikkaamalla voidaan valita Add New (kuva 47).



KUVA 47. Uuden output-kanavan luominen

Outputkanavan ominaisuuksissa (kuva 47) kanavalle määritellään nimi. Nimi on yleensä sama kuin rajapinnan nimi. Speaker typen valinnaksi tulee aina tiedosto. Muita speaker typejä en ole nähnyt koskaan käytettävän, mutta niistä on kerrottu tarkemmin Service Connectin käsikirjassa, joten en käy niitä kuitenkaan tässä yhteydessä läpi. Muista ominaisuuksista kanavan kuvaukseen (description) on ollut käytäntönä kirjoittaa tiedostopolku hakemistoon, johon outputkanava kautta lähetetään tietoja.



KUVA 48. Output-kanavan ominaisuudet ja asetukset

Kuvassa 48 on myös kanavan asetusikkuna, joka aukeaa kanavan ominaisuus ikkunasta löytyvällä Configure-painikkeella. Asetuksissa määritellään tiedostopolku,

johon tiedostot järjestelmästä lähetetään (file path). Mask-kenttään puolestaan määritellään nimi, joka muodostettavalle XML-dokumentille annetaan. Conversion kentässä valitaan lisä-osa, jonka mukaisesti XML-dokumentti muotoillaan. Kaikissa MediusFlow:hun lähetettävissä tiedostoissa Epicorin oma Message Envelope riisutaan, jolloin jäljelle jää vain varsinainen data.

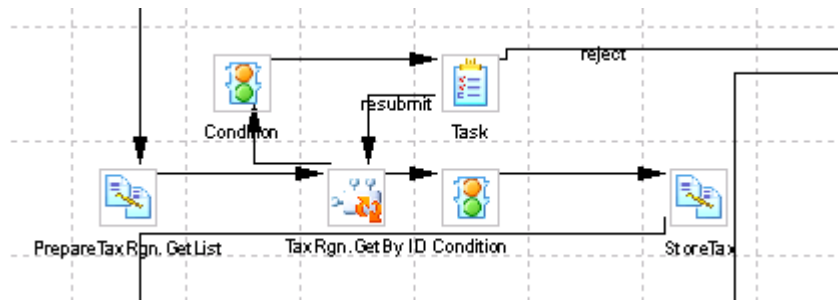
7.2.3 Workflow-prosessi

Uusi workflow luodaan Service Connect Administration Consolessa. Workflow packages-valikossa määritellään ensin paketti, johon tuleva prosessi luodaan. Paketit nimetään yleensä asiakkaan tai E9:ään integroitavan järjestelmän perusteella. E9–Medius-integraation prosessit luodaan Medius-pakettiin. Workflow-paketti vastaa fyysistä kansiota Service Connectin hakemistossa. Kun uusi workflow luodaan, ei siinä ole muuta kuin alku- ja lopetuselementit. Tämä on alkuvaihe, josta prosessia lähdetään työstämään eteenpäin. En tässä yhteydessä käy läpi workflow:n toteutusta, koska se on pitkä prosessi ja vaatisi hyvin yksityiskohtaisen selvityksen sen toteutukseen liittyvistä vaiheista.

Liitteessä 5 on kuvattu Workflow Designerilla rakennettu prosessi kokonaisuudessaan. Workflow Designerin komponentit on käyty läpi jo aiemmin. Prosessin aloituskomponenttiin määritelty schema on Epicorin Vendor.Updaten päivityspyynnön (UpdateRequest) mukainen. UpdateRequestin mukainen schema saadaan usein luomalla Workflow Designerin avulla yksinkertainen prosessi, johon sijoitetaan ainoastaan Poster-komponentti, joka säädetään ohjaamaan tiedot tähän tarkoitettuun, integraatiohakemistosta löytyvään TestOut-kansioon. Tämä prosessi liitetään osaksi asiaan liittyvän liiketoimintaolion päivitysmetodiin, jolloin TestOut-kansioon syntyy järjestelmän muodostama XML-dokumentti tallennetuista tiedoista, jonka jälkeen schema luodaan Schema Utilityn avulla.

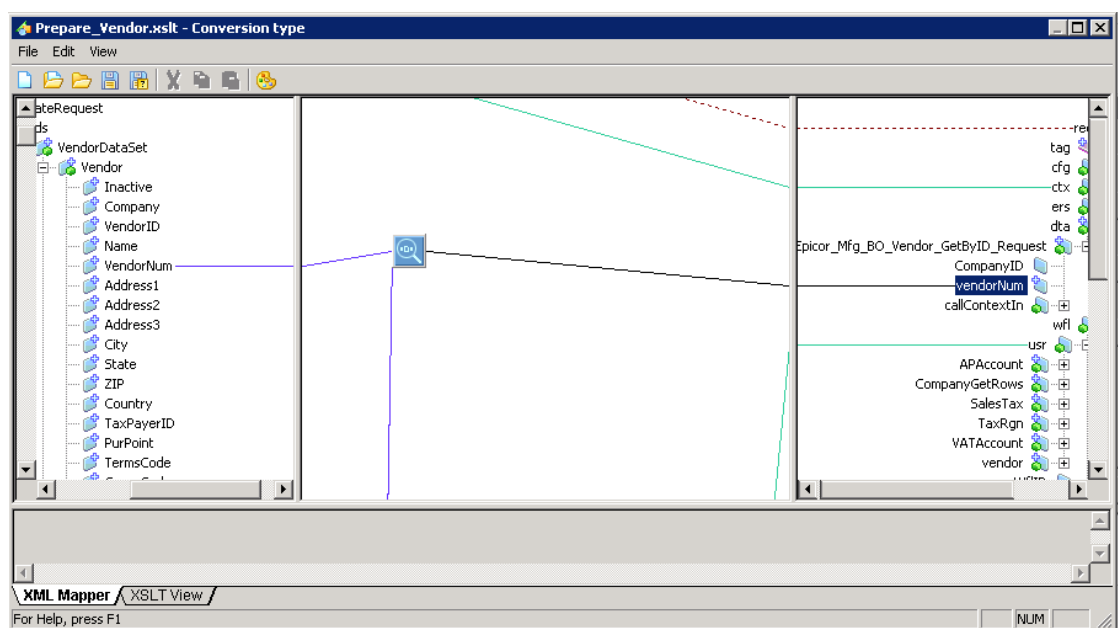
Liitteessä 5 kuvattu prosessi sisältää selkeästi kuusi kokonaisuutta. Näistä kokonaisuuksista viisi on rakennettu .NET-kutsun ympärille. .NET-kutsun avulla voidaan kutsua eri liiketoimintametodeja sen mukaan, mitä tietoa E9:n tietokannasta halutaan hakea. Kokonaisuuden muodostavat NET-komponentti, jonka avulla suoritetaan tietyn liiketoimintametodin kutsu, NET-komponenttia edeltävä XSL

muunnos, jonka avulla välitetään parametrit .NET-kutsun suorittamalle liiketoimintametodille, .NET komponentin jälkeen suoritettava XSL-muunnos, jonka tehtävänä on varastoida liiketoimintaolion hakutulokset niiden myöhempää käyttöä varten prosessin aikana, Condition-komponentit. Näiden avulla tutkitaan, onnistuuko liiketoimintametodin kutsu ja task-komponentti, joka suorittaa liiketoimintametodin uudestaan, mikäli se jostain syystä epäonnistuu.



KUVA 49. M1_E9_Medius_Supplier koostuu viidestä .NET-komponentin ympärille rakennettua kokonaisuutta

Kuvassa 49 on esitetty hyvä esimerkki tällaisesta kokonaisuudesta, joka M1_E9_Medius_Supplier –prosessiin on toteutettu. Nuolet kuvaavat tiedon kulkua prosessissa eri toimintojen välillä. Ensimmäisen muunnoksen avulla välitetään parametrit sitä seuraavalle liiketoimintametodin kutsulle (kuva 50).



KUVA 50. Parametrien välitys liiketoimintametoodeille

Workflow Designerin avulla luodun prosessin lopuksi poster-komponentti suorittaa XML-dokumentin lähettämisen integraatiohakemistossa sijaitsevaan kansioon. Tätä ennen prosessin aikana kerätyt tiedot yhdistellään XSL-muunnoksen avulla haluttuun muotoon rajapintakohtaisten XML schemojen (kuva 40) perusteella.

Prosessin onnistuneen suorittamisen lopputuloksena pitäisi syntyä XML-dokumentti, jossa on käyttäjän tallentamat tiedot sellaisessa muodossa, että se kelpaa myös MediusFlow:hun, eikä prosessin aikana synny mitään virheitä. Esimerkki tällaisesta XML-dokumentista liitteessä

7.2.4 MsgMedius_Supplier.xml

Käyttäjän E9:ssä tallentamat tiedot täytyy vielä lukea Mediukseen. Tästä osuudesta vastaa Medius Integration Gateway –prosessi, kuten aiemmin on jo tullut esille. Medius Integration Gateway on huomattavasti kevyemmän tason integraatiosovellus. Prosessit rakennetaan MIG:n käyttöohjeiden mukaisesti ja prosesseissa suoritettavat tehtävät määritellään XML-dokumenteissa, jotka tallennetaan tiettyyn hakemistoon. Tässä kappaleessa tarkoitukseni on kuvailla prosessi, joka lukee Service Connect-prosessissa syntyneen XML-dokumentin sisältämät toimittajan tiedot ja kirjoittaa ne MediusFlow:n integraatiotauluun, joka tässä tapauksessa on siis NAV_SUPPLIER.

Tätä integraatiota varten toteutetut Medius Integration Gatewayn message-prosessit suorittavat seuraavat toiminnot:

- Integraatiotaulun tyhjennys TRUNCATE TABLE –komennolla.
- Tiedoston luku integraatiohakemistosta
- Tiedon rakenteen validointi XSD-tiedoston perusteella ja muuntaminen
- Tiedon kirjoitus (INSERT INTO) asiaankuuluviin NAV-tauluihin
- USP_StdTransferInformation komennon suoritus, joka puolestaan suorittaa joukon muita komentoja. Nämä komennot lukevat tiedot NAV-tauluista ja välittää loppullisiin tauluihin, joita MediusFlow käyttää
- Alkuperäisen XML-dokumentin siirtäminen arkistointihakemistoon.

Toimittajien tiedot M1_E9_Medius_Supplier-kansiosta hakee MsgMedius_Supplier-niminen prosessi. Nimeämiskäytäntö muotoutui jo alkuvaiheessa nykyisen kaltaiseksi, eikä sitä ole sen jälkeen muutettu. Prosessin nimestä toisaalta voidaan tulkita selkeästi, että kyseessä on integraatiossa nimenomaan Medius Integration Gatewayn message-tyypin prosessi. Nimen loppuosa määräytyy prosessin käsittelemän tiedon mukaan.

7.3 Muut rajapinnat

Edellisessä kappaleessa käsitellyn M1-rajapinnan lisäksi integraatioon kuuluu joukko muita rajapintoja, jotka on toteutettu samojen menetelmien avulla kuin M1-rajapinta. Jokaista perustietoa kohden on toteutettu oma rajapintansa, jotta tiedot olisivat aina ajantasalla MediusFlow:ssa. Ainoa suurempi ero rajapintojen välillä on yleensä Service Connectin avulla luodussa workflow-prosessissa, koska jotkin perustiedot ovat jakaantuneet useampaan eri tietokantatauluun Epicorin tietokannassa. Tämän vuoksi prosessin aikana on jouduttu kutsumaan eri liiketoimintametodeja näiden tietojen hakemiseksi. M1_E9_Medius_Supplier-prosessi on yksi hyvä esimerkki tällaisesta tapauksesta, jossa workflow-prosessin aikana on jouduttu suorittamaan peräti kuusi eri liiketoimintametodin kutsua, jotta kaikki tarvittavat tiedot saadaan välitettyä Epicorista MediusFlow:hun. Vastaavasti osa perustietorajapinnoista sisältää ainoastaan muunnoksen ennen tietojen postausta poster-elementin kautta integraatiohakemistoon.

8 PÄÄTÄNTÖ

Teknologiat kehittyvät nopeasti eikä järjestelmien väliset integraatiot ole teknisesti vaikeita toteuttaa. Pitkälle kehitetyt integraatiosovellukset, kuten Epicor Service Connect, tarjoavat tehokkaat työkalut järjestelmäintegraatioiden toteuttamiselle. Nykyaikaisia teknologioita käyttäen ja palvelinkoneiden suorituskyvyn parantuessa kyetään toteuttamaan yhä monimutkaisempia järjestelmiä, jotka kykenevät vastaamaan yrityksen liiketoiminnan kannalta tärkeisiin vaatimuksiin.

Suurempi haaste on järjestelmien sovittaminen liiketoiminnan prosesseihin. Yrityksen toimintatavoissa ja toimintaan liittyvissä prosesseissa voi olla paljon sellaisia yksityiskohtia, joiden vuoksi järjestelmien tulee olla mukautuvaisia tarpeiden mukaan. Tämä vaikuttaa suoraan myös järjestelmäintegraatioihin, sillä yrityksille kehittyy aina omia tapoja tehdä tietyt liiketoimintaan liittyvät asiat. Nämä asiat on usein otettava huomioon myös järjestelmäintegraatioissa, jonka vuoksi yleispätevän integraatoratkaisun toteuttaminen ei ole koskaan täysin mahdollista. Tämä on ollut nähtävissä myös Epicor 9:n ja MediusFlow:n integraatiossa. Tavoitteena oli toteuttaa näiden kahden järjestelmän välinen integraatio siten, että sen voisi tarjota kokonaisuutena asiakkaalle sellaisenaan. Osittain tässä myös onnistuttiin, mutta silti rajapintoihin oli tehtävä asiakaskohtaisia muutoksia jonkin verran.

LÄHTEET

Bendoly, Elliot 2005. Strategic ERP Extension and Use. Palo Alto: Stanford University Press.

Korhonen, Janne J. 2003. Integroinnista tuli orkestrointia. Talentum Lehtiarkisto. WWW-dokumentti. <http://lehtiarkisto.talentum.com>. Luettu 27.3.2012.

Lahti, Sanna & Salminen, Tero 2008. Kohti Digitaalista taloushallintoa – sähköiset talouden prosessit käytännössä. Juva: WS Bookwell Oy.

Linthicum, David S. 2004. Next Generation Application Integration. Upper Saddle River: Pearsons Education, Inc.

Manouvrier, Bernard & Menard, Laurent 2010. Application Integration : EAI, B2B, BPM and SOA. Hoboken: Wiley-ISTE.

North, Peter & Hermans, Paul 2000. XML – Parhaat ratkaisut ja olennaiset taidot. Helsinki: Edita.

Parthasarathy, S. 2007. Enterprise Resource Planning: A Managerial and Technical Perspective. Daryaganj: New Age International.

Puhakka, Aapo 2004. Väliohjelmistojen kehitysnäkymiä. WWW-dokumentti. <http://aapo.iki.fi/valiohjelmistot.html>. Päivitetty 13.5.3004. Luettu 6.3.2012.

Pulkkinen, Matti P. 2005. Palvelu sopii tietoturvaan. Talentum Lehtiarkisto. WWW-dokumentti. <http://lehtiarkisto.talentum.com>. Luettu 2.4.2012.

Talvitie, Harri 2004. Web Services. Talentum Lehtiarkisto. WWW-dokumentti. <http://lehtiarkisto.talentum.com>. Luettu 16.3.2012.

Themistocleous, Marinos 2005. Enterprise Resource Planning And Enterprise Application Integration. Bradford: Emerald Group Publishing Ltd.

Tähtinen, Sami 2005. Järjestelmäintegraatio – Tarve, vaihtoehdot, toteutus. Helsinki: Talentum.

XPath. WWW-dokumentti. World Wide Web Consortium.
<http://www.w3schools.com/xpath/>. Luettu 21.3.2012.

Sähköisen laskun käyttö yrityksissä (Tilastokeskus 2011)

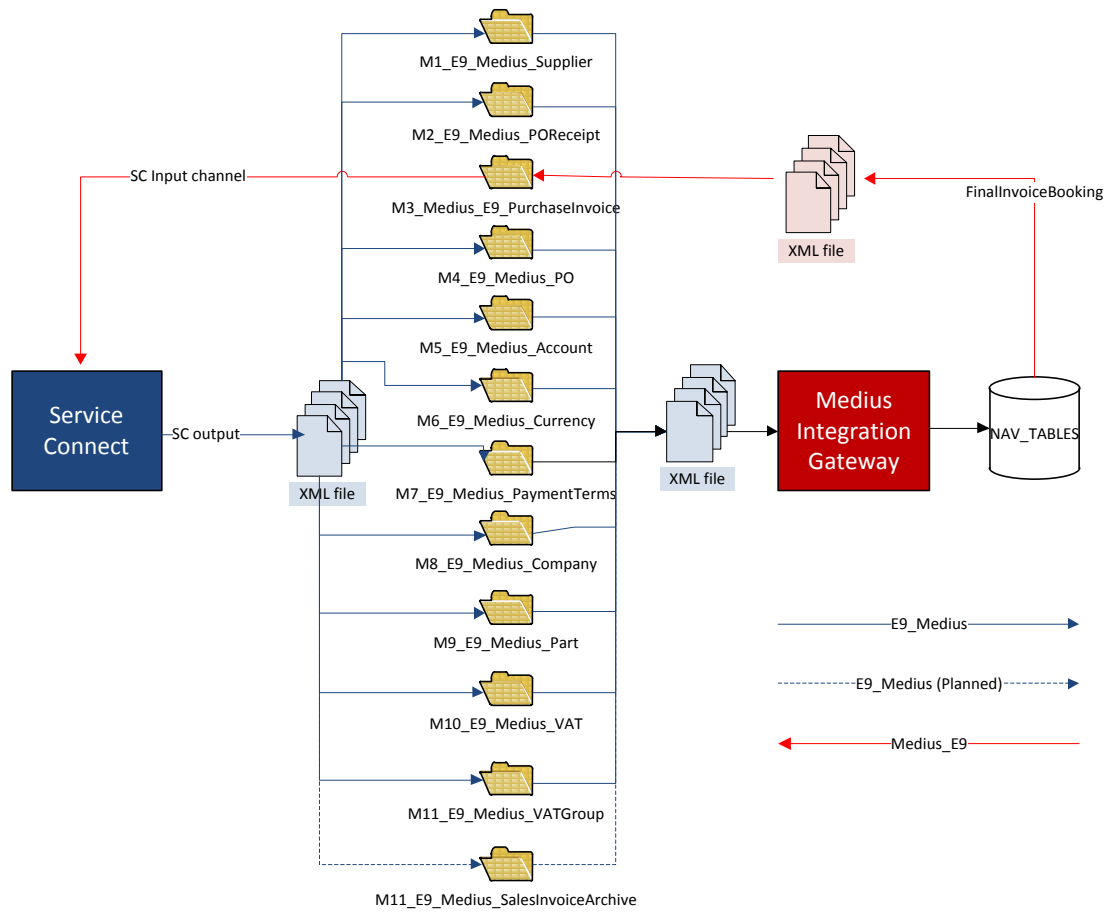
	Sähköisen laskun vastaanotto					Sähköisen laskun lähettäminen				
	Yhteensä 1)	Kehittynyt lasku			Muu, esim. sähkö- posti- lasku	Yhteensä 2)	Kehittynyt lasku			Muu, esim. sähkö- posti- lasku
		Yhteensä	Verkko- lasku	EDI-lasku			Yhteensä	Verkko- lasku	EDI-lasku	
Toimiala	%	%	%	%	%	%	%	%	%	%
Teollisuus	77	45	44	14	65	69	52	46	19	39
Rakentaminen	75	46	45	16	58	57	42	41	9	30
Tukkukauppa	85	41	38	14	77	77	60	52	23	48
Vähittäiskauppa	75	42	39	12	61	45	32	31	7	27
Kuljetus ja varastointi	78	47	47	7	60	66	50	48	6	43
Majoitus- ja ravitsemistoiminta	76	40	38	14	60	47	28	28	2	26
Informaatio ja viestintä	94	70	69	20	83	74	59	59	15	52
Ammatillinen, tieteellinen ja tekninen toiminta	92	62	61	17	73	77	61	58	15	43
Hallinto- ja tukipalvelut	74	48	46	13	64	63	50	50	10	34
Henkilöstön määrä										
10-19	76	41	40	9	64	58	41	40	9	36
20-49	81	49	46	16	67	68	52	47	14	41
50-99	84	57	55	19	70	73	60	55	19	41
100+	91	75	72	34	67	80	72	67	32	36
Kaikki yritykset	79	48	46	14	66	64	49	46	13	38

1) Vastaanottanut vähintään yhdellä sähköisen laskun muodoista.

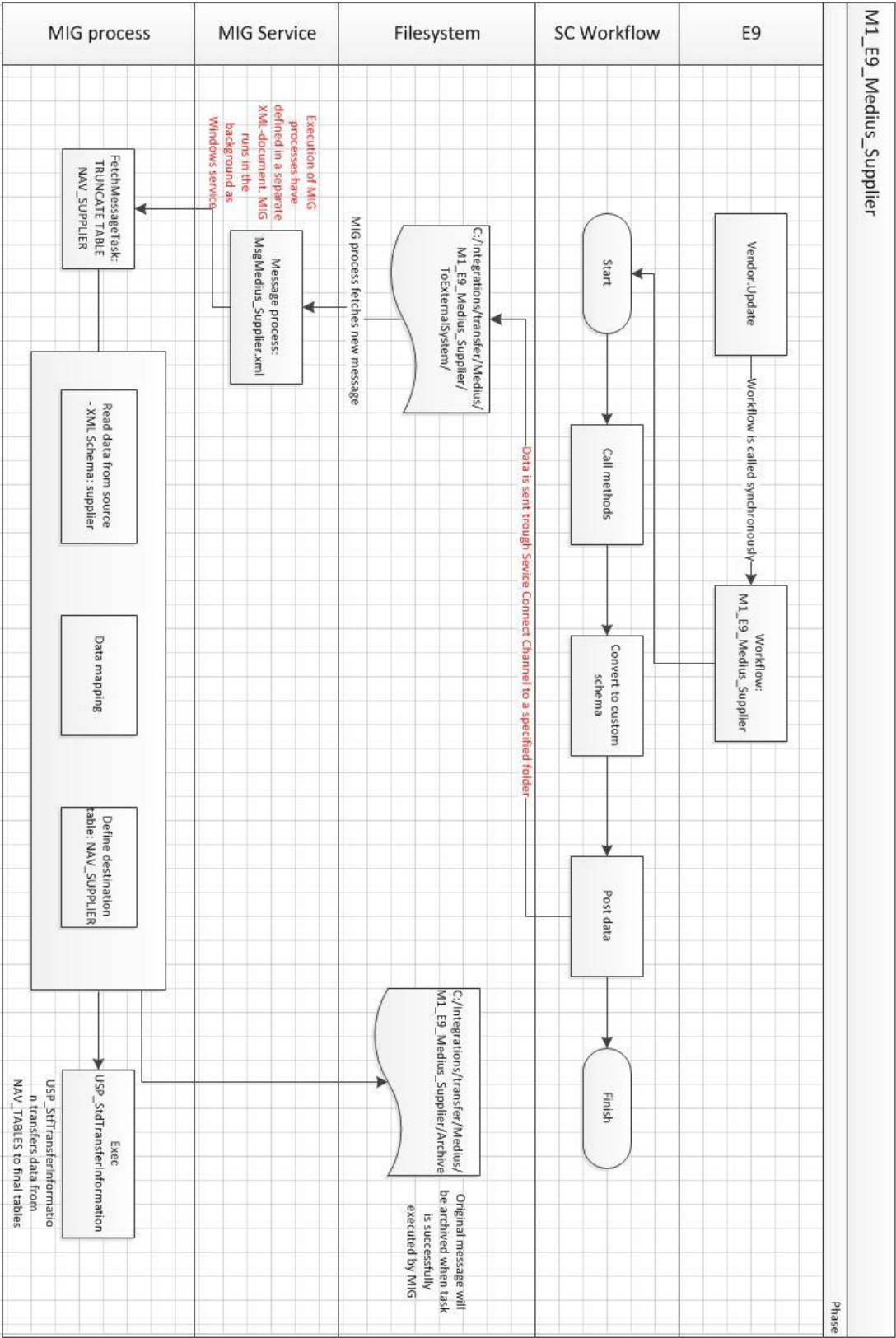
2) Lähettänyt vähintään yhdellä sähköisen laskun muodoista.

Lähde: Tietotekniikan käyttö yrityksissä 2011, Tilastokeskus

Epicor 9:n ja MediusFlow:n välisten rajapintojen yleiskuvaus



Toimittajan tiedon kulku rajapinnassa



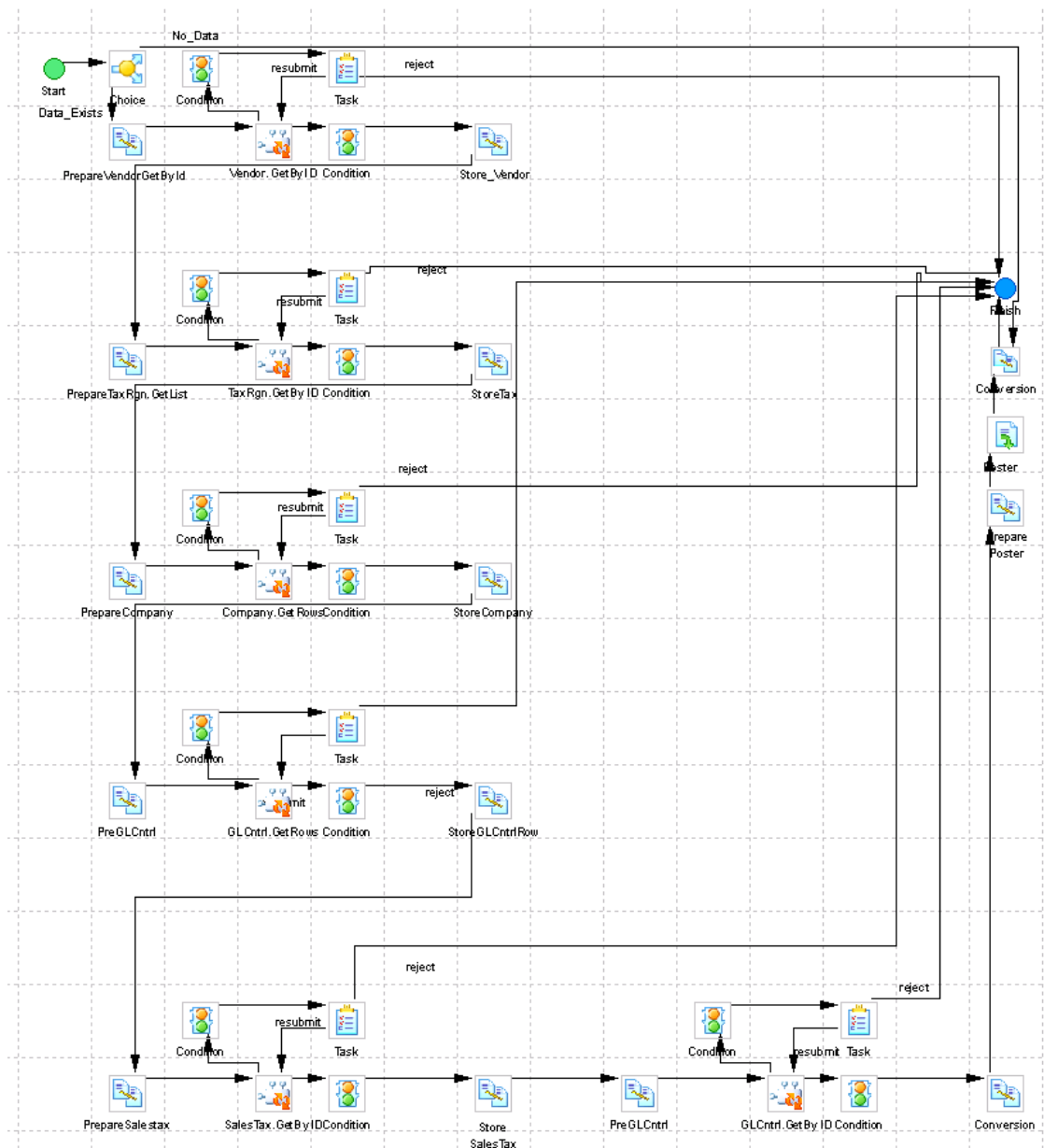
MediusFlown NAV-SUPPLIER –taulu

Taulun kenttä	Kuvaus	Datatyyppi (pituus)	Pakollinen MediusFlow:ssa	Integraatiossa
CompanyId	Yrityksen ID	nvarchar (100)	Kyllä	Kyllä
SupplierId	Toimittajan ID	nvarchar (100)	Kyllä	Kyllä
SupplierId2	2. Toimittajan ID	nvarchar (40)	Ei	Kyllä
Name	Toimittajan nimi	nvarchar(1000)	Kyllä	Kyllä
Name2	2. Toimittajan nimi	nvarchar(1000)	Ei	Kyllä
Address	Toimittajan osoite	nvarchar(1000)	Ei	Kyllä
City	Toimittajan kotipaikkakunta	nvarchar(100)	Ei	Kyllä
Zip	Postinumero	nvarchar(50)	Ei	Kyllä
Country	Toimittajan kotimaa	nvarchar(100)	Ei	Kyllä
Telephone	Puhelinnumero	nvarchar(100)	Ei	Kyllä
Fax	Toimittajan Fax	nvarchar(100)	Ei	Kyllä
Homepage	Toimittajan kotisivut	nvarchar(200)	Ei	Kyllä
Email	sähköpostiosoite	nvarchar(200)	Ei	Kyllä
PaymentTerms	Maksuehdot	nvarchar(100)	Kyllä	Kyllä
OrganizationNumber	Y-tunnus	nvarchar(100)	Ei	Kyllä
VATRegistrationNumber	ALV-tunnus	nvarchar(100)	Ei	Kyllä
EANNumber	EAN-tunnus	nvarchar(100)	Ei	Kyllä
BankAccountNumber	Pankkitili	nvarchar(100)	Ei	Kyllä
PostalAccountNumber	Postitili	nvarchar(100)	Ei	Kyllä
Currency	Valuutta	nvarchar(20)	Kyllä	Kyllä
LedgerAccount	Kirjanpito-tili	nvarchar(100)	Kyllä	Kyllä
CostCenter	Kulupaikka	nvarchar(60)	Ei	Kyllä
CostUnit	Kuluyksikkö	nvarchar(60)	Ei	Kyllä
Project	Projekti	nvarchar(60)	Ei	Kyllä
CostAccount		nvarchar(100)	Ei	Kyllä
CostCostCenter		nvarchar(60)	Ei	Kyllä
CostCostUnit		nvarchar(60)	Ei	Kyllä
CostProject		nvarchar(60)	Ei	Kyllä
VATAccount	ALV-tili	nvarchar(100)	Kyllä	Kyllä
PrelAccount		nvarchar(60)	Ei	Kyllä
PrelCostCenter		nvarchar(60)	Ei	Kyllä
PrelCostUnit		nvarchar(60)	Ei	Kyllä
PrelProject		nvarchar(60)	Ei	Kyllä
PrelCostAccount		nvarchar(60)	Ei	Kyllä
PrelCostCostCenter		nvarchar(60)	Ei	Kyllä
PrelCostCostUnit		nvarchar(60)	Ei	Kyllä
PrelCostProject		nvarchar(60)	Ei	Kyllä
FreightAccount		nvarchar(60)	Ei	Kyllä
FreightCostCenter		nvarchar(60)	Ei	Kyllä
FreightCostUnit		nvarchar(60)	Ei	Kyllä

MediusFlown NAV-SUPPLIER –taulu

FreigthProject		nvarchar(60)	Ei	Kyllä
PackingAccount		nvarchar(60)	Ei	Kyllä
PackingCostCenter		nvarchar(60)	Ei	Kyllä
PackingCostUnit		nvarchar(60)	Ei	Kyllä
PackingProject		nvarchar(60)	Ei	Kyllä
AdjustmentAccount		nvarchar(60)	Ei	Kyllä
AdjustmentCostCenter		nvarchar(60)	Ei	Kyllä
AdjustmentCostUnit		nvarchar(60)	Ei	Kyllä
AdjustmentProject		nvarchar(60)	Ei	Kyllä
EUSupplier	1=kyllä, 0=ei	int(4)	Ei	Kyllä
VAT	VATId in MF	nvarchar(30)	Kyllä	Kyllä
Active	-1=kyllä, 0=ei	nvarchar(70)	Kyllä	Kyllä
ExpenseType	Kulutyyppi	nvarchar(60)	Ei	Kyllä
ResponsibleUser		nvarchar(60)	Ei	Kyllä
InternalSupplier	1 = kyllä, 0= ei	int(4)	Ei	Kyllä
BusGroup		nvarchar(100)	Ei	Kyllä
Trusted		int(4)	Ei	Kyllä
LanguageISO		char(2)	Ei	Kyllä
EnvironmentalAccount		nvarchar(60)	Ei	Ei
EnvironmentalCostCente		nvarchar(60)	Ei	Ei
EnvironmentalCostUnit		nvarchar(60)	Ei	Ei
EnvironmentalProject		nvarchar(60)	Ei	Ei
PalletAccount		nvarchar(60)	Ei	Ei
PalletCostCenter		nvarchar(60)	Ei	Ei
PalletCostUnit		nvarchar(60)	Ei	Ei
PalletProject		nvarchar(60)	Ei	Ei
DeliveryTerm		nvarchar(60)	Ei	Ei
SupplierAdjustmentRequ estAccount		nvarchar(60)	Ei	Ei
SupplierAdjustmentRequ estCostCenter		nvarchar(60)	Ei	Ei
SupplierAdjustmentRequ estCostUnit		nvarchar(60)	Ei	Ei
SupplierAdjustmentRequ estProject		nvarchar(60)	Ei	Ei
SupplierAdjustmentRequ estUnitQuantityAccount		nvarchar(60)	Ei	Ei
ToleranceAccount		nvarchar(60)	Ei	Ei
UnitPriceAccount		nvarchar(60)	Ei	Ei
QuantityAccount		nvarchar(60)	Ei	Ei
FTX_MFLOW_1	Vapaa tekstikenttä	nvarchar(1000)	Ei	Kyllä
FTX_MFLOW_2	Vapaa tekstikenttä	nvarchar(1000)	Ei	Kyllä
FTX_MFLOW_3	Vapaa tekstikenttä	nvarchar(1000)	Ei	Kyllä
FTX_MFLOW_4	Vapaa tekstikenttä	nvarchar(1000)	Ei	Kyllä
FTX_MFLOW_5	Vapaa tekstikenttä	nvarchar(1000)	Ei	Kyllä

M1_E9_Medius_Supplier –workflow



M1_E9_Medius_Supplier –prosessin muodostama XML-dokumentti

```
<?xml version="1.0" encoding="utf-16"?>
<!-- ESIMERKKIDATAA. -->
<Supplier>
    <CompanyId>YRITYS</CompanyId> <!-- Testiyritys. -->
    <SupplierId>TOIMITTAJA</SupplierId>
    <Name>TESTITOIMITTAJA</Name>
    <Name2></Name2>
    <Address></Address>
    <City></City>
    <Zip></Zip>
    <Country></Country>
    <Telephone></Telephone>
    <Fax></Fax>
    <Homepage></Homepage>
    <EMail></EMail>
    <PaymentTerms>N30</PaymentTerms>
    <OrganizationNumber></OrganizationNumber>
    <VATRegistrationNumber></VATRegistrationNumber>
    <EANNumber></EANNumber>
    <BankAccountNumber></BankAccountNumber>
    <PostalAccountNumber></PostalAccountNumber>
    <Currency>EUR</Currency>
    <LedgerAccount>1000</LedgerAccount>
    <CostCenter></CostCenter>
    <CostUnit></CostUnit>
    <Project></Project>
    <CostAccount></CostAccount>
    <CostCostCenter></CostCostCenter>
    <CostCostUnit></CostCostUnit>
    <CostProject></CostProject>
    <VATAccount>2000</VATAccount>
    <PrelAccount></PrelAccount>
    <PrelCostCenter></PrelCostCenter>
```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```
<PrelCostUnit></PrelCostUnit>
<PrelProject></PrelProject>
<PrelCostAccount></PrelCostAccount>
<PrelCostCostCenter></PrelCostCostCenter>
<PrelCostCostUnit></PrelCostCostUnit>
<PrelCostProject></PrelCostProject>
<FreightAccount></FreightAccount>
<FreightCostCenter></FreightCostCenter>
<FreightCostUnit></FreightCostUnit>
<FreightProject></FreightProject>
<PackingAccount></PackingAccount>
<PackingCostCenter></PackingCostCenter>
<PackingCostUnit></PackingCostUnit>
<PackingProject></PackingProject>
<AdjustmentAccount></AdjustmentAccount>
<AdjustmentCostCenter></AdjustmentCostCenter>
<AdjustmentCostUnit></AdjustmentCostUnit>
<AdjustmentProject></AdjustmentProject>
<EUSupplier></EUSupplier>
<VAT>VAT</VAT> <!-- VAT? -->
<Active>-1</Active>
<ExpenseType></ExpenseType>
<ResponsibleUser></ResponsibleUser>
<InternalSupplier></InternalSupplier>
<BusGroup></BusGroup>
<Trusted></Trusted>
<LanguageISO></LanguageISO>
<FTX_MFLOW_1></FTX_MFLOW_1>
<FTX_MFLOW_2></FTX_MFLOW_2>
<FTX_MFLOW_3></FTX_MFLOW_3>
<FTX_MFLOW_4></FTX_MFLOW_4>
<FTX_MFLOW_5></FTX_MFLOW_5>
</Supplier>
```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

<?xml version="1.0" encoding="utf-16"?>
<Message xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ConfigParameters />
  <MessageName>MsgMedius_Supplier.xml</MessageName> <!--Prosessin nimi-->
  <MessageType>Primary</MessageType> <!-- Aina Primary -->
  <Version>
    <Number>1.0.0.0</Number>
    <Standard>false</Standard>
  </Version>
  <Comment />
  <FetchMessageTask> <!-- Ennen varsinaista prosessia suoritettavat toiminnot -->
    <ConnectionType>Database</ConnectionType>
    <ConnectionSubType>0</ConnectionSubType>
    <ConnectionString>
      <!-- Avataan tietokantayhteys -->
      data source=[tietokantapalvelin];
      initial catalog=[tietokanta];
      integrated security=False;
      persist security info=False;
      workstation id=[tyoaseman_nimi];
      user id=[kayttaja];
      password=[salasana];
      packet size=4096</ConnectionString>

    <Action>
      <!-- Tyhjennetään NAV_SUPPLIER, jottei yritetä kirjoittaa samaa tietoa
      Näin vältetään virheet
      -->
      Truncate Table NAV_SUPPLIER;
    </Action>
    <Remote />
    <Parameters />
  </FetchMessageTask>
  <NewMessageSource>
    <SourceId>FileHandler</SourceId> <!-- MIG:n messageadapteri. Lisätietoja MIG:n ohjeista -->
    <Remote />
    <ConnectionType>File</ConnectionType> <!-- Yhteystyyppi. Vertaa tässä Service Connectin Speaker Typen valinta -->
    <ConnectionSubType>3</ConnectionSubType> <!-- Connection subtype 3 = XML Multiple. Lisätietoja XML
    MIG:n ohjeista. -->
    <ConnectionString>
      <!-- tiedostopolku, josta tiedostot luetaan -->
      C:\integrations\transfer\Medius\M1_E9_Medius_Supplier\toExternalSystem\
    </ConnectionString>
    <SelectionStatement>GETALLFILES</SelectionStatement>
    <Multidimensional>false</Multidimensional>
    <Parameters>
      <Parameter>
        <Name>FileExtension</Name>
        <Value>*.xml</Value>
        <UseLookUp>false</UseLookUp>
      </Parameter>
      <Parameter>
        <Name>OnlyFirst</Name>
        <Value>False</Value>
        <UseLookUp>false</UseLookUp>
      </Parameter>
    </Parameters>
    <StaticColumns />
    <Functions />
    <Filter />
  </NewMessageSource>
  <SourceMessageLines>
    <Source>
      <SourceId>Supplier</SourceId>
      <Remote />
      <ConnectionType>XML</ConnectionType>
      <ConnectionSubType>3</ConnectionSubType>
      <ConnectionString>
        <!-- tiedostopolku kansioon, jossa luettavan XML-dokumentin rakenne
        on kuvattu. Huom: Tämä on sama XSD-tiedosto kuin mitä workflow-
        prosessissa käytetään. -->
        C:\Medius_E9\medius_schemas\supplier.xsd
      </ConnectionString>
      <SelectionStatement>C:\integrations\transfer\Medius\M1_E9_Medius_Supplier\toExternalSystem\</SelectionStatement>
      <Multidimensional>true</Multidimensional>
      <Parameters>
        <Parameter>
          <Name>FileExtension</Name>

```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

        <Value>*.xml</Value>
        <UseLookUp>false</UseLookUp>
    </Parameter>
</Parameters>
<StaticColumns />
<Functions />
<Filter />
</Source>
</SourceMessageLines>
<Mappings>
<!--
Mappings-tagien väliin määritellään mappaukset samalla tavalla kuin ne määritellään XML Mapperin avulla
Service Connectissa. Koska XML-dokumentit muunnetaan jo Service Connectissa, tarvitsee MIG-prosessi
mapata tiedot "i=1" -periaatteella. (CompanyId=CompanyId)
-->
<Mapping>
    <SourceId>Supplier.Supplier</SourceId>
    <DestinationId>NAV_SUPPLIER</DestinationId>
    <MappingValues>
        <MappingValue>
            <Source>CompanyId</Source>
            <Destination>CompanyId</Destination>
        </MappingValue>
        <MappingValue>
            <Source>SupplierId</Source>
            <Destination>SupplierId</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Name</Source>
            <Destination>Name</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Name2</Source>
            <Destination>Name2</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Address</Source>
            <Destination>Address</Destination>
        </MappingValue>
        <MappingValue>
            <Source>City</Source>
            <Destination>City</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Zip</Source>
            <Destination>Zip</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Country</Source>
            <Destination>Country</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Telephone</Source>
            <Destination>Telephone</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Fax</Source>
            <Destination>Fax</Destination>
        </MappingValue>
        <MappingValue>
            <Source>Homepage</Source>
            <Destination>Homepage</Destination>
        </MappingValue>
        <MappingValue>
            <Source>EMail</Source>
            <Destination>EMail</Destination>
        </MappingValue>
        <MappingValue>
            <Source>PaymentTerms</Source>
            <Destination>PaymentTerms</Destination>
        </MappingValue>
        <MappingValue>
            <Source>OrganizationNumber</Source>
            <Destination>OrganizationNumber</Destination>
        </MappingValue>
        <MappingValue>
            <Source>VATRegistrationNumber</Source>
            <Destination>VATRegistrationNumber</Destination>
        </MappingValue>
    </MappingValues>
</Mapping>

```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

<MappingValue>
  <Source>EANNumber</Source>
  <Destination>EANNumber</Destination>
</MappingValue>
<MappingValue>
  <Source>BankAccountNumber</Source>
  <Destination>BankAccountNumber</Destination>
</MappingValue>
<MappingValue>
  <Source>PostalAccountNumber</Source>
  <Destination>PostalAccountNumber</Destination>
</MappingValue>
<MappingValue>
  <Source>Currency</Source>
  <Destination>Currency</Destination>
</MappingValue>
<MappingValue>
  <Source>LedgerAccount</Source>
  <Destination>LedgerAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>CostCenter</Source>
  <Destination>CostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>CostUnit</Source>
  <Destination>CostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>Project</Source>
  <Destination>Project</Destination>
</MappingValue>
<MappingValue>
  <Source>CostAccount</Source>
  <Destination>CostAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>CostCostCenter</Source>
  <Destination>CostCostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>CostCostUnit</Source>
  <Destination>CostCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>CostProject</Source>
  <Destination>CostProject</Destination>
</MappingValue>
<MappingValue>
  <Source>VATAccount</Source>
  <Destination>VATAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelAccount</Source>
  <Destination>PrelAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelCostCenter</Source>
  <Destination>PrelCostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelCostUnit</Source>
  <Destination>PrelCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelProject</Source>
  <Destination>PrelProject</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelCostAccount</Source>
  <Destination>PrelCostAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelCostCostCenter</Source>
  <Destination>PrelCostCostCenter</Destination>
</MappingValue>

```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

<MappingValue>
  <Source>PrelCostCostUnit</Source>
  <Destination>PrelCostCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>PrelCostProject</Source>
  <Destination>PrelCostProject</Destination>
</MappingValue>
<MappingValue>
  <Source>FreightAccount</Source>
  <Destination>FreightAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>FreightCostCenter</Source>
  <Destination>FreightCostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>FreightCostUnit</Source>
  <Destination>FreightCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>FreightProject</Source>
  <Destination>FreightProject</Destination>
</MappingValue>
<MappingValue>
  <Source>PackingAccount</Source>
  <Destination>PackingAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>PackingCostCenter</Source>
  <Destination>PackingCostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>PackingCostUnit</Source>
  <Destination>PackingCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>PackingProject</Source>
  <Destination>PackingProject</Destination>
</MappingValue>
<MappingValue>
  <Source>AdjustmentAccount</Source>
  <Destination>AdjustmentAccount</Destination>
</MappingValue>
<MappingValue>
  <Source>AdjustmentCostCenter</Source>
  <Destination>AdjustmentCostCenter</Destination>
</MappingValue>
<MappingValue>
  <Source>AdjustmentCostUnit</Source>
  <Destination>AdjustmentCostUnit</Destination>
</MappingValue>
<MappingValue>
  <Source>AdjustmentProject</Source>
  <Destination>AdjustmentProject</Destination>
</MappingValue>
<MappingValue>
  <Source>EUSupplier</Source>
  <Destination>EUSupplier</Destination>
</MappingValue>
<MappingValue>
  <Source>VAT</Source>
  <Destination>VAT</Destination>
</MappingValue>
<MappingValue>
  <Source>Active</Source>
  <Destination>Active</Destination>
</MappingValue>
<MappingValue>
  <Source>ExpenseType</Source>
  <Destination>ExpenseType</Destination>
</MappingValue>
<MappingValue>
  <Source>ResponsibleUser</Source>
  <Destination>ResponsibleUser</Destination>
</MappingValue>
<MappingValue>
  <Source>InternalSupplier</Source>
  <Destination>InternalSupplier</Destination>
</MappingValue>

```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

    <MappingValue>
      <Source>BusGroup</Source>
      <Destination>BusGroup</Destination>
    </MappingValue>
    <MappingValue>
      <Source>Trusted</Source>
      <Destination>Trusted</Destination>
    </MappingValue>
    <MappingValue>
      <Source>LanguageISO</Source>
      <Destination>LanguageISO</Destination>
    </MappingValue>
    <MappingValue>
      <Source>FTX_MFLOW_1</Source>
      <Destination>FTX_MFLOW_1</Destination>
    </MappingValue>
    <MappingValue>
      <Source>FTX_MFLOW_2</Source>
      <Destination>FTX_MFLOW_2</Destination>
    </MappingValue>
    <MappingValue>
      <Source>FTX_MFLOW_3</Source>
      <Destination>FTX_MFLOW_3</Destination>
    </MappingValue>
    <MappingValue>
      <Source>FTX_MFLOW_4</Source>
      <Destination>FTX_MFLOW_4</Destination>
    </MappingValue>
    <MappingValue>
      <Source>FTX_MFLOW_5</Source>
      <Destination>FTX_MFLOW_5</Destination>
    </MappingValue>
  </MappingValues>
</Mapping>
</Mappings>
<Destination>
  <!-- Määränpää määritellään Destination-tagien väliin
        Adapterina database. Tarkempia lisätietoja MIG:n ohjeissa -->
  <DestinationId>Supplier</DestinationId>
  <Remote />
  <ConnectionType>Database</ConnectionType>
  <ConnectionSubType>0</ConnectionSubType>
  <ConnectionString><!-- Avataan tietokantayhteys -->
    data source=[tietokantapalvelin];
    initial catalog=[tietokanta];
    integrated security=False;
    persist security info=False;
    workstation id=[tyoaseman_nimi];
    user id=[kayttaja];
    password=[salasana];
    packet size=4096
  </ConnectionString>
  <DestinationPath>
    <!-- Aina multitable E9-MediusFlow -integraatiossa-->
    MultiTable
  </DestinationPath>
  <Parameters>
    <Parameter>
      <Name>Tables</Name>
      <Value>NAV_SUPPLIER</Value>
      <UseLookUp>false</UseLookUp>
    </Parameter>
    <Parameter>
      <!-- ConvertByContent = false, jottei MIG muunna esimerkiksi ID-tietojen
            datatyyppejä numeroiksi, jos ID-tiedot ovat pelkkiä numeroita
            ja päinvastoin -->
      <Name>ConvertByContent</Name>
      <Value>False</Value>
    </Parameter>
  </Parameters>
</Destination>

```

MIG-prosessin asetukset –MsgMedius_Supplier.xml

```

<TasksOnErrorReply />
<TasksOnSuccessReply>
  <!-- Suoritettavat tehtävät, mikäli varsinaisen prosessin lopputuloksena
        luetun ja käsitellyn XML-dokumentin tiedot on viety NAV-tauluun -->
  <Task>
    <!-- Suoritetaan MediusFlow:n tietokannassa oleva
        Stored Procedure: Exec USP-StdTransferInformation-->
    <ConnectionType>Database</ConnectionType>
    <ConnectionSubType>0</ConnectionSubType>
    <ConnectionString><!-- Avataan tietokantayhteys -->
        data source=[tietokantapalvelin];
        initial catalog=[tietokanta];
        integrated security=False;
        persist security info=False;
        workstation id=[tyoaseman_nimi];
        user id=[kayttaja];
        password=[salasana];
        packet size=4096
    </ConnectionString>
    <Action>Exec USP_StdTransferInformation</Action>
    <Remote />
    <Parameters />
  </Task>
  <Task>
    <!-- Arkistoidaan käsitelty XML-dokumentti sen jälkeen, kun se on käsitelty onnistuneesti -->
    <ConnectionType>File</ConnectionType>
    <ConnectionSubType>3</ConnectionSubType>
    <ConnectionString>C:\integrations\transfer\Medius\M1_E9_Medius_Supplier\toExternalSystem\</ConnectionString>
    <Action>Move</Action>
    <Remote />
    <Parameters>
      <Parameter>
        <Name>ToFolder</Name>
        <Value>C:\integrations\transfer\Medius\M1_E9_Medius_Supplier\Archive\</Value>
        <UseLookUp>false</UseLookUp>
      </Parameter>
    </Parameters>
  </Task>
</TasksOnSuccessReply>
<TasksOnUnhandledException />
</Message>

```