Tomi Virta

Scriben jonopohjaisten integraatioprosessien käyttöönotto ja automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

4.5.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Tomi Virta Scriben jonopohjaisten integraatioprosessien käyttöönotto ja automatisointi 39 sivua + 0 liitettä 4.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Yliopettaja Auvo Häkkinen Lehtori Jussi Alhorinne Simo Syrjänen

Insinöörityön tarkoituksena oli ottaa käyttöön Scriben jonopohjaiset integraatioprosessit Talentumilla tehtävän dataintegraation suorituskyvyn ja luotettavuuden parantamiseksi. Tavoitteena oli selvittää suorituskykytestien avulla, kuinka paljon jonojen käyttö paransi suorituskykyä. Lisäksi tavoitteena oli automatisoida jonopohjaisia integraatioprosesseja. Työn toimeksiantaja oli Talentum Oyj.

Työssä tutustuttiin ensiksi Talentumilla käytössä oleviin asiakkuudenhallintajärjestelmiin nimeltä Tabu ja Microsoft Dynamics CRM. Seuraavaksi perehdyttiin Scribe Insight –integraatiotyökalun komponentteihin ja ominaisuuksiin. Tämän jälkeen tutustuttiin Scriben jonopohjaisten integraatioprosessien toimintaan ja niiden käyttöönoton vaatimuksiin.

Seuraavaksi jonopohjaiset integraatioprosessit otettiin käyttöön ja niiden suorituskykyä testattiin kolmella eri prosessilla: Asiakkaiden päivitys, tilausten päivitys ja laskujen päivitys. Asiakas-testissä saavutettiin 5-kertainen siirtonopeus Tabun ja CRM:n välillä. Tilaus-testissä saavutettiin 2-kertainen nopeus ja lasku-testissä 5,45 kertainen nopeus. Suorituskykyjen erot johtuivat prosessien monimutkaisuuksien eroista sekä CRM:n rajoitteista. Testeillä saavutettujen tulosten perusteella jonopohjaiset integraatioprosessit paransivat suorituskykyä.

Testien aikana ilmenneet ongelmat osoittivat, että jonojen käyttö vaatii erilaisia toimenpiteitä, joita olisi myös mahdollista automatisoida jatkossa. Esimerkiksi Scriben lokitaulu täytyy puhdistaa säännöllisesti, jottei se pääse täyttymään ja samalla hidastamaan Scriben toimintaa.

Lopuksi kolme testattua prosessia automatisoitiin päivittäin ajettaviksi. Lopputuloksena oli kolme jonopohjaista automatisoitua integraatioprosessia.

Avainsanat	Scribe Insight, MSMQ, CRM

Author(s) Title Number of Pages Date	Tomi Virta Implementation and Automation of Scribe Queue Based Integration Processes 39 pages + 0 appendices 4 May 2016
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	Software Engineering
Instructor(s)	Auvo Häkkinen, Principal Lecturer Jussi Alhorinne, Senior Lecturer Simo Syrjänen

The purpose of this thesis was the implementation of Scribe Insight's queue integration processes to improve the performance and reliability of Talentum's data integrations. The goal was to perform a series of performance tests and find out how the queue based integration processes affected performance. In addition the aim was to automate said integration processes. The thesis was commissioned by Talentum Oyj.

The first part of the thesis focuses on customer relationship management software called Tabu and Microsoft Dynamics CRM which are currently used by Talentum. The next part takes a look at the components and features of data integration software called Scribe Insight. After this the thesis focuses on the functionality and requirements of the queue based integration processes provided by Scribe.

Then the thesis describes how the queue integration processes were implemented and how their performance was tested. The performance was tested with three processes: customer update, order update and invoice update. A five times better performance between Tabu and CRM was achieved in the customer test. The order test resulted in two times better performance and the invoice test resulted in 5.45 times better performance. The differences in performance were due to differences in the complexity of the processes and the limitations of CRM. The queue integration processes improved the performance based on the results of the tests.

The problems encountered during testing indicated that the use of queues requires different measures to be taken. These measures can also be automated in the future. For example, the Scribe log-table must be cleaned regularly to prevent it from getting filled up and slowing down Scribe in the process.

Finally the three processes were automated to be run daily. The end results were three automated queue integration processes.

Keywords	Scribe Insight, MSMQ, CRM

Sisällys

Lyhenteet

ActiveMQ	Avoimen lähdekoodin Java-pohjainen viestinvälittäjä
CRM	Customer Relationship Management eli asiakkuudenhallinta.
ESB	Enterprice Service Bus on tietojärjestelmäarkkitehtuurin malli, jota käytetään sovellusten välisen kommunikaation suunnitteluun ja implementointiin
ETL	Extract Transform Load on prosessi, jossa lähteestä haettu data muunnetaan haluttuun muotoon ja viedään kohteeseen.
MSMQ	Microsoft Message Queuing viestintäprotokolla.
SMTP	Simple Mail Transfer Protocol on sähköpostiviestien välittämiseen käytetty protokolla.
SOAP	Simple Object Access Protocol on Microsoftin kehittämä tietoliikenneprotokolla, joka mahdollistaa proseduurien etäkutsun.
SQL	Structured Query Language on relaatiotietokantojen kyselyihin käytettävä kieli.

1 Johdanto

Talentum Oyj on luopumassa Tabu-nimisestä asiakkuudenhallintajärjestelmästä. Tabussa pidetään yllä Talentumin tuotteiden, kuten kirjojen ja lehtien, sekä asiakkaiden tietoja. Lisäksi Tabussa säilytetään asiakkaiden tilaus- ja laskutustietoja.

Luopumisen yhteydessä Tabun tietokannasta halutaan siirtää miljoonia rivejä dataa toiseen yrityksessä käytössä olevaan asiakkuudenhallintajärjestelmään nimeltä Microsoft Dynamics CRM (Customer Relationship Management). Datan integraatio tapahtuu Scribe Insight -nimisellä ETL-työkalulla (Extract Transform Load). Scribe on datan integroimiseen tarkoitettu ohjelmisto, jolla pystyy helposti graafisen käyttöliittymän avulla tekemään integraatioprosesseja esimerkiksi tietokantojen välille. Tällaisia prosesseja on jo tehty Tabun ja CRM:n välille, mutta ne ovat liian hitaita suurten massojen integroimiseen.

Tämän opinnäytetyön tavoitteena on nopeuttaa Tabun ja CRM:n välisiä dataintegraatioita ottamalla käyttöön Scriben säikeistetyt jonopohjaiset integraatioprosessit. Koska kumpikin järjestelmä on tuotantokäytössä ja data elää koko ajan, on tavoitteena myös automatisoida näitä integraatioprosesseja, jotta data pysyisi eheänä järjestelmien välillä.

Opinnäytetyössä perehdytään asioihin, jotka ovat keskeisiä jonopohjaisten integraatioprosessien käyttöönotossa. Työstä on rajattu pois asioita, jotka saattavat olla keskeisiä integraation kannalta, mutta eivät ole keskeisiä säikeistyksen käyttöönotossa kuten yksityiskohtaiset kuvaukset tietokantojen tauluista. Työssä tutustutaan ensin järjestelmiin, joiden välille nykyinen integraatio on rakennettu. Tarkoituksena on perehtyä sekä Tabun että CRM:n ominaisuuksiin ja raijoitteisiin, jotka vaikuttavat integraatioon. Seuraavaksi tutustutaan Scribeen ja sen tuomiin mahdollisuuksiin ja ominaisuuksiin. Tämän jälkeen käydään läpi nykyisten integraatioprosessien logiikka. Sitten on tarkoituksena perehtyä Microsoft Message Queuing viestintä protokollan toimintaan, johon Scriben jonopohjaiset integraatioprosessit perustuvat. Seuraavaksi perehdytään Scriben jonopohjaisiin prosesseihin ja niiden tuomiin hyötyihin.

Tämän jälkeen otetaan jonopohjaiset integraatioprosessit käyttöön ja kuvataan tarvittavat toimenpiteet. Käyttöönoton valmistuttua tehdään joukko suorituskykytestejä ja ana-

lysoidaan sekä vertaillaan niiden tuloksia säikeettömiin integraatioprosesseihin. Tuloksena saadaan tieto, kuinka paljon säikeistys vaikuttaa suorituskykyyn. Käyttöönoton ja testien jälkeen automatisoidaan prosessit päivittäin ajettaviksi, jotta järjestelmän välillä liikkuva data pysyy ajan tasalla. Lopuksi analysoidaan jonopohjaisten prosessien käyttöönoton tuomia hyötyjä ja ilmenneitä haittoja. Lisäksi selvitetään, minkälaisia jatkotoimenpiteitä ja ylläpitoa jonopohjaisten integraatioprosessien käyttö vaatii.

2 Asiakkuudenhallintajärjestelmät

Tässä luvussa tutustutaan Talentumilla käytössä oleviin asiakkuudenhallintajärjestelmiin nimeltä Tabu ja Microsoft Dynamics CRM. Luvussa käydään läpi Tabun ja CRM:n käyttötarkoituksia sekä ominaisuuksia. Lisäksi tutustutaan Tabun ja CRM:n väliseen nykyiseen integraatioon sekä syihin, miksi Tabusta halutaan luopua.

2.1 Tabu

Tabu on Talentumilla 1990-luvun lopusta käytössä ollut tilaus-, laskutus- ja reskontrajärjestelmä. Se on Tietoenator-nimisen yrityksen (nykyään Tieto) koodaama sovellus, joka on käytössä ainoastaan Talentumilla. Alun perin Tabu suunniteltiin lehtiliiketoimintaa varten tilaus- ja toimitusjärjestelmäksi, mutta vuonna 2005 siihen lisättiin tarvittava toiminallisuus kirjaliiketoiminnalle. Vuonna 2001 Tabun kannasta jouduttiin tekemään duplikaattikanta POP-onlinetilausten hallintaa varten, koska Tabu itsessään ei taipunut siihen.

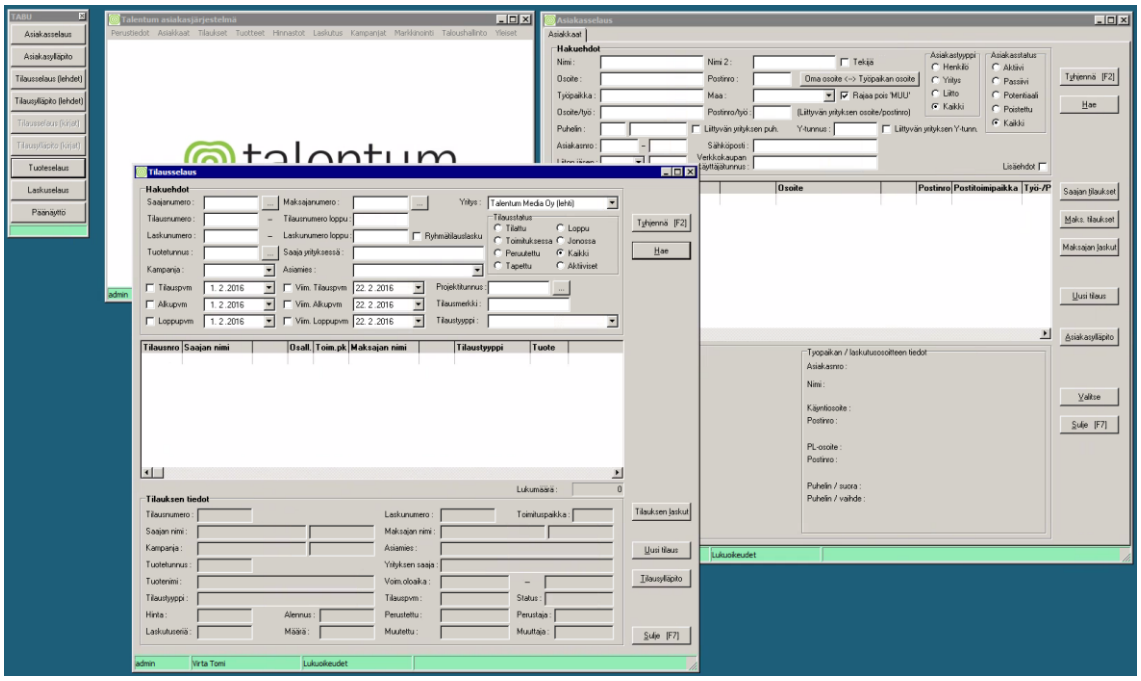
Tabussa pidetään yllä tietoja asiakkaista, tuotteista, tilauksista sekä laskuista. Lisäksi Tabu sisältää tietoja Talentumin tekemästä markkinoinnista. Tabussa hoidetaan myös tilausten jaksotus ja asiakkaiden yhdistäminen yrityksiin.

Tabun tietokantana toimii Oraclen relaatiotietokannan versio 8.1.7.0.0, joka julkaistiin 2002 marraskuussa. Tietokannasta löytyy yli 3 miljoonan asiakkaan tiedon, 7 tuhatta tuotetta, 7,4 miljoonaa tilausta ja 4,4 miljoonaa laskua sekä kymmeniä miljoonia rivejä markkinointitietoja.

Tabun pääasiallinen käyttö jakautuu lehdille ja kirjoille. Kirjapuolella Tabua käytetään pääasiassa tilausten ja laskujen hallintaan, sekä asiakas- ja tuotetietojen ylläpitoon. Lehtipuolella pidetään näiden lisäksi yllä hinnastotietoja, ilmestymisaikatauluja, kampanjoita

sekä markkinointitietoja. Tabusta tehdään myös kaikki tarvittavat markkinointipoiminnat lehtiliiketoimintaa varten. Tabuun avataan lisäksi suoramyyntikampanjoita, jotta asiakaspalvelu pystyy tallentamaan tilaukset oikeille kampanjoille. Kampanjoiden hallinta ja markkinointi tapahtuu kuitenkin pääasiassa CRM:ssä.

Tabusta halutaan luopua useasta syystä. Ensinnäkin se on kallis ylläpitää ja kehittää eikä sen tietoihin voi aina luottaa. Esimerkiksi tilausten jaksotukset eivät aina noudata kirjanpidon periaatteita. Lisäksi sen käyttöliittymä (kuva 1) on vanhanaikainen eikä erityisen luonteva uusille käyttäjille.



Kuva 1. Tabun käyttöliittymä

Kuvasta 1 voi jo huomata, ettei sitä suunniteltu käyttäjäkeskeisesti. Vain yhden asiakkaan tietoja voi katsoa kerrallaan ja uusia ikkunoita avautuu melkein joka tilanteessa, kun haluaa tarkempaa tietoa esimerkiksi laskuista tai tilauksista. Tämä aiheuttaa lisää vaivaa uusien käyttäjien koulutuksessa. Tabun yhteensopivuus on myös heikko muiden järjestelmien kanssa, eikä se anna mahdollisuuksia rajapintojen käyttöön. Tästä syystä käytön laajentaminen on lähes mahdotonta, eikä Talentumilta löydy siihen tarvittavaa osaamista.

2.2 Microsoft Dynamics CRM 2011

Microsoft Dynamics CRM on toinen Talentumilla käytössä oleva asiakkuudenhallintajärjestelmä. Sen tietokantana toimii SQL Server 2008 R2. Tietokanta sisältää kaikkien Talentumin liiketoimintayksiköiden asiakastiedot.

Vuonna 2010 Talentum osti IIR Finland Oy -nimisen yrityksen, jonka mukana tuli Brainsniminen asiakastietokanta. Tässä vaiheessa Talentumilla oli kolme asiakastietokantaa käytössä: Tabu, POP ja Brains. Talentum halusi keskitetyn asiakastietokannan, josta voisi tehdä ristiin markkinointipoimintoja näiden kantojen välillä. Tähän tarkoitukseen vuonna 2011 Talentum hankki CRM:n. Käyttöönotossa toimi konsulttina Fenix Solutions Oy.

2.3 CRM-Integraatio

Kolmesta käytössä olevasta asiakastietokannasta ensimmäisenä aloitettiin Tabun integraatio CRM:ään. Talentum hankki Cybercom-nimiseltä yhtiöltä mittatilaustyönä tehdyt SOAP pohjaiset ESB-sovellukset Tabun, POPin ja Brainsin integraatioihin. Tämä maksoi satoja tuhansia euroja.

Sovellukset veivät datan ActiveMQ-jonon avulla CRM:n www-sovelluspalvelun läpi CRM:n kantaan. Kun Tabu oli saatu integroitua, seurasi POP-kanta ja viimeisenä Brains. Tässä vaiheessa kaikki data oli saatavilla CRM:n kannasta.

Brains ja POP poistuivat käytöstä integraation myötä. Tabu jäi edelleen käyttöön, sillä CRM:stä ei löytynyt tarvittavaa toiminallisuutta, jota Tabu tarjosi. Integraatioiden jäljiltä CRM:ään oli tullut suuri määrä duplikaatteja johtuen Brainsin ja Tabun välisistä yhtäläisyyksistä.

3 Integraatiotyökalu Scribe Insight

Scribe Insight on datan integroimiseen tarkoitettu ETL-työkalu (Extract Transform Load), jolla pystyy helposti graafisen käyttöliittymän avulla tekemään tiedon siirtoja erilaisten sovellusten tai tietokantojen välillä. Scribellä voi suorittaa luku-, päivitys-, kirjoitus-, muunnos- ja poisto-operaatioita erätyyppisinä tai reaaliaikaisina ajoina. Se sisältää erilaisia adaptereja, joista tärkeimpänä tämän työn kannalta on CRM-adapteri, jonka avulla Scribe saa helposti siirrettyä dataa CRM:ään. Lisäksi työssä käytetään Oracle- ja SQL server -adaptereja, joiden avulla saadaan haettua dataa sekä Tabun että CRM:n kannasta.

Ennen Scriben käyttöönottoa Talentumilla käytettiin integraatioihin Cybercomin tekemiä ESB-sovelluksia. Näiden muokkaaminen oli kuitenkin hidasta ja kallista. Jos haluttiin uusia attribuutteja, muutokset piti tehdä suoraan sovellukseen. Siihen tarvittiin aina ulkopuolisia tekijöitä, jotka maksoivat kymmeniä tuhansia euroja kuukaudessa. Vuonna 2013 otettiin käyttöön Scribe Insight, joka mahdollisti integraatioiden tekemisen sisäisesti ilman ulkopuolista koodausapua. Talentumilla käytössä oleva Scribe Standard -lisenssi maksaa noin 20 000 € vuodessa. Se sisältää yhden testipalvelinlisenssin ja mahdollistaa kahdeksan viestiprosessorin (engl. message processor) käytön sekä toimii enintään 250 CRM-käyttäjän kanssa. Yksi viestiprosessori on käytännössä yksi säie.

Scribe Insight koostuu kahdesta ylätason komponentista sekä sisäisestä tietokannasta. Nämä komponentit ovat Scribe client tools ja Scribe Integration server. Client tools jakautuu vielä kahdeksi komponentiksi: Scribe Workbenchiksi ja Scribe Consoleksi. Tässä työssä kaikki komponentit ovat käytössä.

3.1 Scribe Integration Server

Scribe Integration Server eli integraatiopalvelin koostuu kaikista Scriben palveluista, joita ovat AdminServer, BridgeServer, Eventmanager, MessageServer ja MonitorServer. Integraatiopalvelimella ei itsellään ole käyttöliittymää, vaan se tarjoaa palvelut, jotka mahdollistavat integraatioprosessien ajamisen ja hallinnoimisen. Kaikki Scriben toiminnot kulkevat integraatiopalvelimen kautta.

Scriben AdminServer mahdollistaa järjestelmän hallinnoinnin sekä paikallisen tai etäyhteyden Scriben palveluihin. BridgeServer siirtää käyttäjän määrittelemiä XML-viestejä Scriben input-jonoon. MessageServer käsittelee nämä XML-viestit ja tekee niille käyttäjän määrittelemiä toimenpiteitä. Eventmanager havaitsee ja hallinnoi tapahtumia, sekä mahdollistaa niiden aikataulutuksen ja automaation. MonitorServer muodostaa virheilmoituksia ja mahdollistaa niiden automaattisen lähettämisen sähköpostitse.

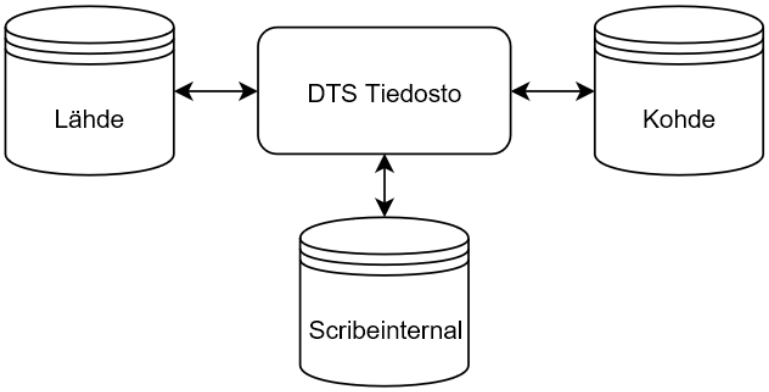
3.2 Scribe Workbench

Scribe Workbench on Scriben pääkäyttöliittymä, jossa kaikki datan integraatiot toteutetaan. Graafisen käyttöliittymän avulla käyttäjä pystyy helposti yhdistämään tietokantoihin sekä määrittelemään lähteen ja kohteen väliset data mappaukset. Lisäksi käyttäjä pystyy määrittelemään datalle tehtävät operaatiot ja niiden suoritusjärjestyksen.

Workbench muodostaa käyttäjän määrittelemistä yhteyksistä ja datamappauksista sekä operaatioista Data Transformation Specification -tiedoston (DTS), jota voidaan myöhemmin käyttää Scribe Consolen kautta. Worbenchin käyttöliittymän kautta pystyy myös testaamaan ja ajamaan yksittäisiä DTS-tiedostoja.

3.2.1 Yhteydet ja adapterit

DTS-tiedoston muodostaminen Workbenchillä alkaa yhteyksien määrittämisellä. Minimivaatimus DTS-tiedoston muodostamiseen on yhteys lähteeseen ja kohteeseen sekä Scriben sisäiseen tietokantaan (kuva 2) (Beckner 2015, 15).



Kuva 2. Scriben vaatimat yhteydet lähteeseen, kohteeseen ja Scribeinternal-tietokantaan.

Lähteenä ja kohteena voi toimia mikä tahansa tietokanta, CSV- tai XML-tiedosto. Myös Scriben sisäinen tietokanta voi toimia lähteenä. Tässä työssä käytetään Scriben omaa CRM-adapteria datan syöttämiseen CRM:ään, sekä Oracle-ajuria datan hakemiseen Tabusta. Lisäksi käytössä on SQL Server -ajuri, jonka avulla on mahdollista tehdä mukautettuja SQL-kyselyjä CRM-kantaan.

Yhteyksien muodostamisen jälkeen täytyy määritellä lähde sekä kohde. Lähdettä määritellessä Workbench näyttää valitun yhteyden sisältä löytyvät taulut ja objektit, joista käyttäjä voi valita haluamansa lähteeksi. Vaihtoehtoisesti käyttäjä voi kirjoittaa mukautetun SQL-kyselyn, jonka palauttamaa dataa voidaan käyttää lähteenä.

3.2.2 Operaatioaskelten määrittäminen

Kohdetta määritellessä käyttäjän täytyy määritellä operaatiot, jotka Workbench suorittaa. Operaatioita voidaan tehdä jokaiselle lähderiville yksi tai useampia ja niille voi määritellä ehtoja, joiden täyttyessä niitä suoritetaan. Tämä mahdollistaa monimutkaistenkin logiikoiden rakentamisen. Käyttäjän täytyy määrittää vähintään yksi operaatioaskel, jotta kohde tulee käyttöön. Operaatioiden kuvaukset löytyvät taulukosta 1.

Taulukko 1. Scriben operaatiot

Operaatio	Toiminto	Selitys
Insert	Lisää lähderivin kohteeseen	Yrittää lisätä tietueen. Jos kyseinen tietue löytyy jo kohteesta, muodostuu duplikaatti tai vaihtoehtoisesti annetaan virheilmoitus
Update	Päivittää kohderivin lähteestä tulevalla datalla	Vaatii hakuehdon, jonka perusteella yritetään päivittää kohteessa oleva tietue.
Update/Insert	Päivittää/Lisää lähderivin kohteeseen	Koostuu kahdesta toiminnosta: Update ja Insert. Yrittää ensin tehdä päivityksen. Jos ei onnistu, lisätään rivi.
Insert/Update	Lisää/Päivittää lähderivi kohteeseen	Sama kuin Update/Insert, mutta päinvastaisessa järjestyksessä
Upsert	Päivittää/Lisää lähderivin kohteeseen.	Samanlainen kuin Update/Insert operaatio, mutta koostuu vain yhdestä operaatiosta. Haittapuolena on huonompi yhteensopivuus
Seek	Selvittää onko tietue olemassa kohteessa	Etsii hakuehdoilla tietueita kohteesta
Delete	Poistaa tietueen kohteesta	Jos mahdollista, poistaa tietueen

Operaatioaskelille voidaan määrittää, mitä tehdään eri tilanteissa kuten virheen sattuessa tai operaation onnistuessa. Oletusarvona operaation onnistuessa transaktion muutokset vahvistetaan ja siirrytään seuraavalle riville. Virheen sattuessa transaktion muutokset perutaan, kirjataan virhe lokiin ja jatketaan seuraavalle riville. Kuolettavissa virheissä lopetetaan toiminnot täysin ja kirjataan virhe lokiin. Esimerkiksi yhteyden katkeaminen on kuolettava virhe.

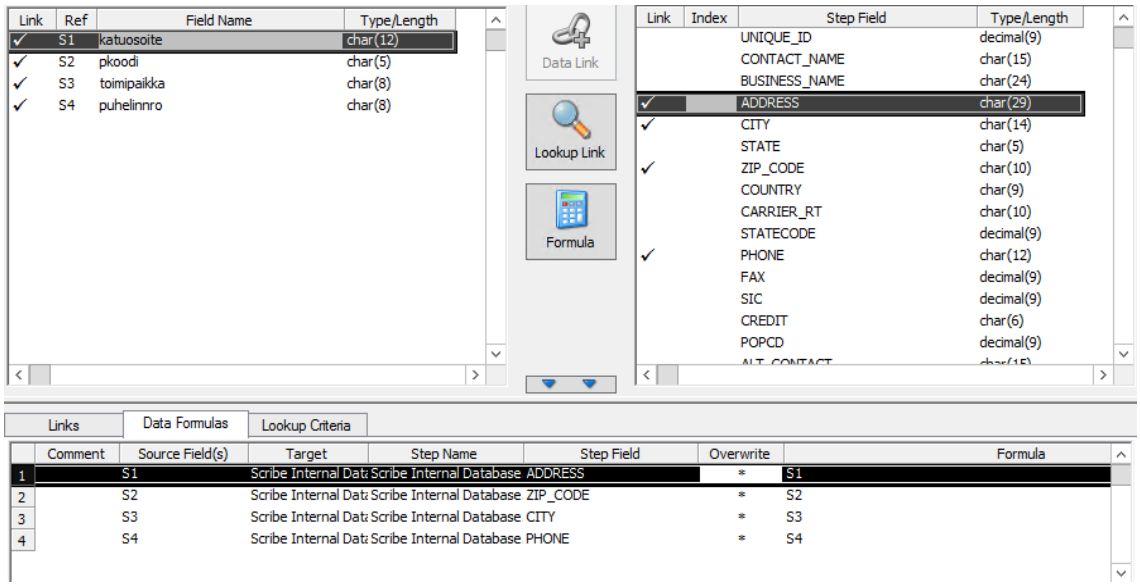
Lisäksi askelille voidaan määritellä Pre-Operation Step Flow Control Formula eli ennen operaatioaskelta suoritettava funktio. Tämän avulla käyttäjä voi esimerkiksi määritellä, missä tilanteissa operaatioaskelta ei suoriteta. Tässä työssä tätä käytetään esimerkiksi asiakkaiden verifioinnin tarkastukseen. Jos asiakas on verifioitu CRM:ssä, sitä ei haluta päivittää. Tehdään siis funktio, joka tarkastaa ennen operaatioaskelta verifionnin. Jos verifiointi löytyy, hypätään operaatioaskeleen yli ja siirrytään seuraavan asiakkaan prosessointiin.

Käyttäjä pystyy myös sallimaan monen tietueen etsintä- ja päivitys-operaatiot. Tämä mahdollistaa monen tietueen päivittämisen samojen hakuehtojen täyttyessä. Kyseinen

asetus on kuitenkin oletusarvoisesti pois päältä, ja se on myös järkevää pitää näin. Tällöin Scribe antaa virheilmoituksen löytäessään samoilla hakuehdoilla monta tietuetta. Näin saadaan kiinni potentiaaliset duplikaatit kohteessa. Se on hyvä ominaisuus varsinkin, jos tiedetään, että duplikaatteja ei saisi olla.

3.2.3 Tietojen mappaus

Lähteen ja kohteen määrittämisen jälkeen voidaan aloittaa tietojen mappaus eli kuvataan, mikä attribuutti lähteestä kohdistuu mihinkin kenttään kohteessa. Tämä tapahtuu helposti Workbenchin graafisen käyttöliittymän avulla (kuva 3).



Kuva 3. Workbenchin tietojen mappaus

Tietojen mappaus jakautuu kahdenlaisiin linkkeihin: data-linkkiin ja lookup-linkkiin. Data-linkki kertoo Scribelle, mihin kenttään lähteestä tuleva data sijoitetaan kohteessa. Lookup-linkin avulla voidaan tehdä seek- tai update-operaatioita. Se etsii lähteestä saamansa arvon avulla vastaavan tietueen kohteesta. Linkkeihin on mahdollista lisätä myös logiikkaa Scriben sisäisillä funktioilla. Scribe tarjoaa suuren määrän erilaisia funktioita, kuten loogisia ja aggregaattifunktioita. Myös käyttäjän määrittelemiä muuttujia voi käyttää linkkien arvoina. Update, update/insert, insert/update ja upsert sekä seek operaatiot tarvitsevat kaikki vähintään yhden lookup-linkin toimiakseen. Muut operaatiot tarvitsevat vain data-linkin.

3.2.4 DTS-tiedoston ajaminen

DTS-tiedoston voi ajaa suoraan Workbenchistä tai ottaa käyttöön Scriben konsolissa heti, kun käyttäjä on määritellyt yhteydet, lähteen ja kohteen sekä tarvittavat tietojen mappaukset. Workbenchin kautta on myös mahdollista testata DTS-tiedoston ajamista. Se simuloi ajoa tietue kerrallaan ja näyttää operaatioaskelten tulokset. Ajon käynnistyessä Workbench näyttää reaaliajassa ajoon kuluneen ajan sekä operaatioiden, virheiden ja ylihypättyjen operaatioiden määrän. Käyttäjä pystyy myös asettamaan päälle seurannan, joka näyttää prosentteina ajon etenemisen. Tämä lisää käynnistämisaikaa, koska Scriben täytyy hakea rivien määrä valmiiksi ennen ajoa.

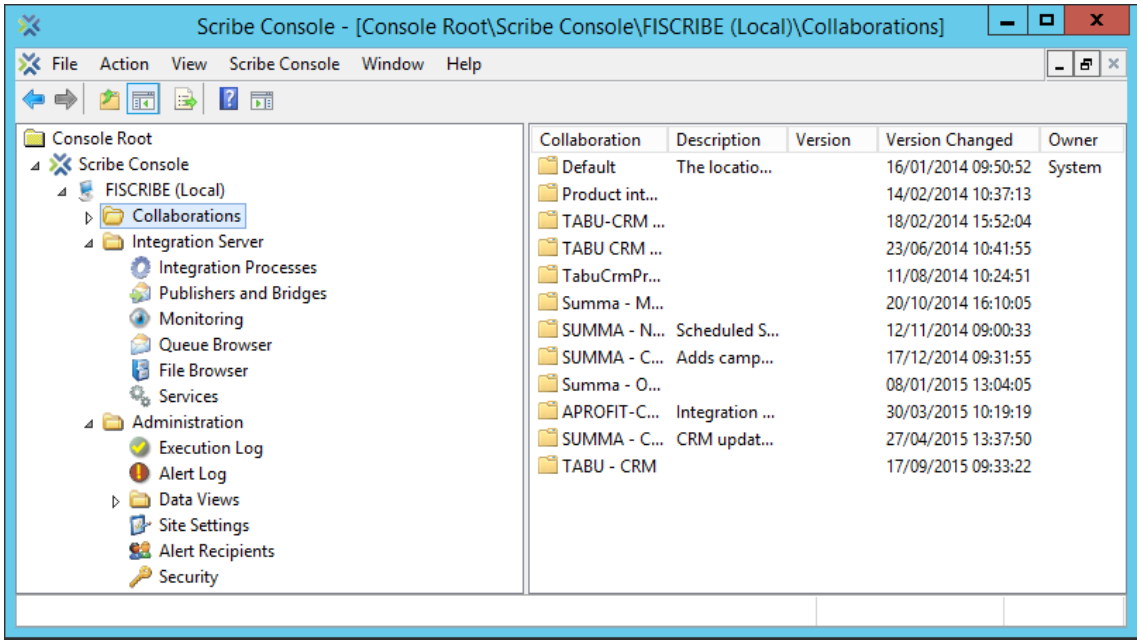
Ajon päätyttyä näytölle tulee yhteenveto ajosta ja mahdollisuus tarkastaa mahdolliset virheet lokista. Lisäksi ajosta voidaan generoida raportti, jossa näkyy yleistä tietoa kuten onnistuneiden ja epäonnistuneiden operaatioiden määrän, alku- ja loppuhetken sekä suorituskykytietoja.

3.2.5 Vikatilanteiden käsittely

Scribe kirjaa automaattisesti sisäiseen kantaansa kaikki epäonnistuneet rivit ja epäonnistumisten syyt. Workbenchissä on lisäksi toiminto nimeltä Rejected Source Rows, joka insertoi lähteestä tulleet hylätyt tietueet käyttäjän määrittelemään tietokantaan. Jotta tätä toimintoa voitaisiin käyttää, täytyy ensin määritellä yhteys kantaan, jota halutaan käyttää. Tämän jälkeen DTS-asetuksista voidaan valita kyseinen kanta ja taulu, johon epäonnistuneet tietueet syötetään. Scribe pystyy myös tarvittaessa luomaan uuden taulun tätä varten. Tämän jälkeen voidaan esimerkiksi luoda toinen DTS-tiedosto, joka käyttää lähteenä epäonnistuneiden rivien taulua ja poistaa lähderivejä käsiteltyään ne onnistuneesti. Näin saadaan tarkasteltua epäonnistuneita tietueita ja ajettua niitä uudelleen.

3.3 Scribe Console

Scribe Console (kuva 4) on yksi Scribe Insightin päätyökaluista.



Kuva 4. Scribe Consolen käyttöliittymä

Se on Microsoft Management Console -pohjainen sovellus, joka toimii puumaisen käyttöliittymän avulla. Console mahdollistaa monimutkaisten integraatioprosessien luonnin, ajastamisen ja automatisoinnin. Lisäksi Console tarjoaa joukon Scriben hallinnointiin liittyviä toimintoja. Consolen toiminallisuus koostuu kolmesta ylätasosta: Collaborations, Integration Server ja Administration.

3.3.1 Koosteet

Scriben koosteet (engl. collaboration) koostuvat integraatioprosesseista, niihin liittyvistä tiedostoista, monitoreista ja data näkymistä. Koosteita käytetään integraatioprosessien organisoimiseen ja ryhmittelyyn. Koosteen alta pääsee käsiksi yksittäisiin integraatioprosesseihin ja niiden tiedostoihin, sekä prosessien monitoreihin ja lokeihin. Tässä työssä tehtävät prosessit ovat yhden koosteen alla nimeltä TABU – CRM.

Collaborations-valikosta käyttäjä pystyy manuaalisesti keskeyttämään, jatkamaan, ajamaan tai poistamaan yksittäisiä integraatioprosesseja tai vaihtoehtoisesti keskeyttämään ja jatkamaan kaikki koosteen alta löytyvät prosessit. Tässä tapauksessa keskeyttäminen ei peruuta jo prosessoituja rivejä vaan lopettaa prosessoinnin. Prosessia ei ole mahdollista jatkaa keskeytyksen jälkeen siitä, mihin jäätiin, vaan se alkaa aina uudelleen alusta.

3.3.2 Integraatioprosessit

Integraatioprosessit ovat Scriben konsolissa luotavia prosesseja, jotka havaitsevat erilaisia tapahtumia ja tapahtumien seurauksena ajavat automaattisesti käyttäjän määrittelemän DTS-tiedoston. Integraatioprosesseja on neljää tyyppiä:

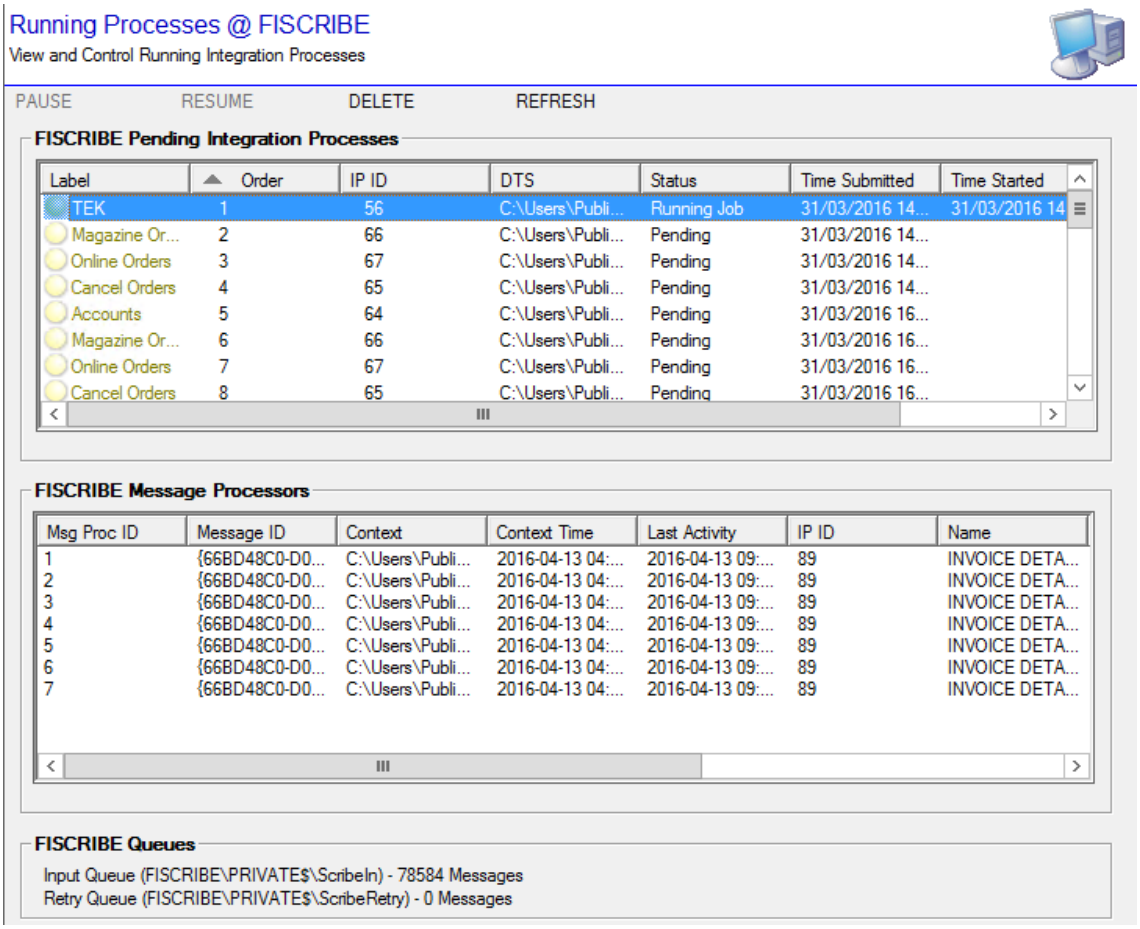
- file
- time
- query
- queue.

File-tyyppinen eli tiedostopohjainen prosessi ajaa DTS-tiedoston havaitessaan tiedoston määritellyssä tiedostopolussa. DTS voidaan ajaa väliajoin niin kauan kuin tiedosto on hakemistossa tai vaihtoehtoisesti kerran havaitsemisen jälkeen. Time-tyyppinen eli ajastettu prosessi ajaa DTS-tiedoston määriteltynä ajankohtana tai tietyin väliajoin. Väliajaksi voi valita mitä tahansa minuuteista kuukauteen. Query-tyypinen eli kyselypohjainen prosessi ajaa DTS:n saadessaan dataa DTS:n omasta lähteestä. DTS voidaan ajaa kerran tai joka kerta, kun lähteestä saadaan dataa. Queue-tyypinen eli jonopohjainen prosessi havaitsee Scriben jonoon saapuvat viestit ja ajaa DTS-tiedoston käyttäen näitä viestejä lähteenä. Tässä työssä tarkoituksena on ottaa käyttöön juuri XML-viestejä havaitsevia jonopohjaisia integraatioprosesseja.

3.3.3 Integration Server

Integration Server -valikkoa käytetään palvelujen ja prosessien konfigurointiin ja hallinnointiin. Täältä käyttäjä pääsee käsiksi kaikkiin integraatioprosesseihin, monitoreihin, lokeihin, palveluihin ja tiedostoihin sekä lisäksi myös jonoihin.

Integration server -näkymästä (kuva 5) saa hyvän yleiskuvan päällä olevista integraatioprosesseista, jonojen viestien lukumäärästä ja viestintäprosessorien toiminnasta.



Kuva 5. Integration Server -näkymä

Kuvan ottamishetkellä oli yksi TEK-niminen integraatio prosessi aktiivisena. Lisäksi seitsemän viestiprosessoria prosessoivat Invoice Detail-nimisiä viestejä. Input-jonossa oli 78584 viestiä.

3.3.4 Monitorit

Monitorien tarkoituksena on lähettää varoituksia määriteltyjen ehtojen täyttyessä. Niiden avulla käyttäjä pystyy helposti seuraamaan integraatioprosessien kulkua ja Scriben antamia virheilmoituksia. Ennen monitorien luomista on määriteltävä Scriben sähköpostiasetukset sekä kenelle monitorien muodostamia varoituksia lähetetään. Sähköpostiasetuksia voi muokata Site settings -lehden alta. Monitoreja varten täytyy määritellä vähintään SMTP-asetukset eli postipalvelimen nimi, lähettäjän sähköpostiosoite ja lähettäjän nimi.

Varoitusten vastaanottajien määrittäminen tapahtuu Scriben konsolin kautta Administration-lehden alta Alert Recipients -kohdasta. Sieltä käyttäjä pystyy lisäämään yksittäisiä sähköposteja ja ryhmittelemään niitä kokonaisuuksiksi, joille varoituksia lähetetään. Lisäksi vastaanottajille voidaan määritellä, minkä tyyppisiä varoituksia heille lähetetään.

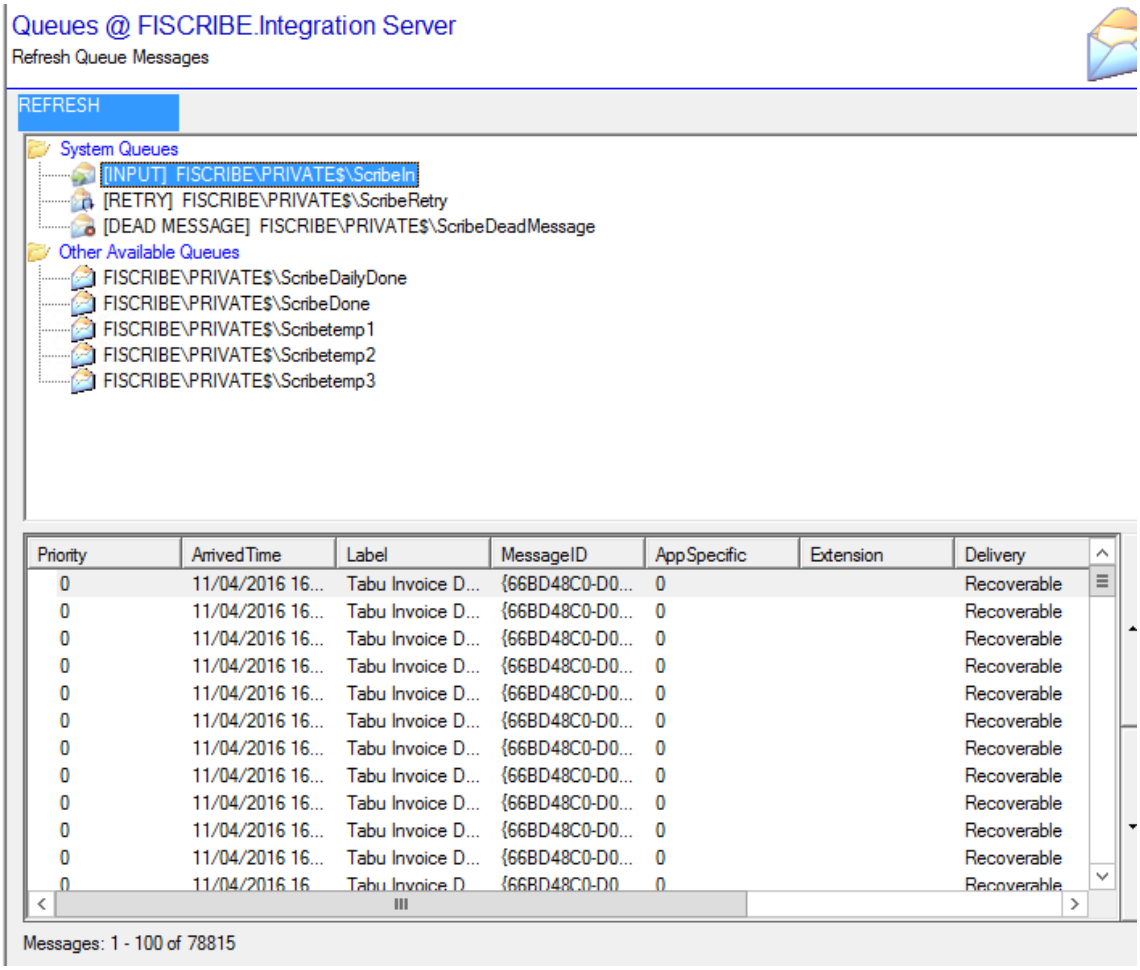
Monitorien luominen tapahtuu Scriben konsolin kautta kuudessa vaiheessa. Ensin määritellään monitorin tyyppi ja nimi. Monitori voi olla joko Query- tai Queue-tyyppiä. Queue tyypin monitorit seuraavat Scriben jonoja ja jonoissa tapahtuvien muutosten perusteella lähettävät varoituksia käyttäjille. Query-tyypin monitorit saavat monitoroitavan datan käyttäjän määrittelemästä SQL-kyselystä. Seuraavaksi määritellään jono tai yhteys mitä monitori tarkkailee riippuen monitorin tyypistä. Kolmannessa vaiheessa määritellään kriteerit, joiden täyttyessä monitori lähettää varoituksen. Kriteereinä voi olla esimerkiksi tietyn viestien määrän ylittyminen jonossa tai jonkin attribuutin arvo. Tässä vaiheessa määritellään myös kenelle varoitus lähtee. Tämän jälkeen asetetaan ajastus. Monitorin ajon aikaväliksi voi valita mitä tahansa minuuteista kuukausiin. Ajastuksen jälkeen voidaan vielä asettaa monitori aktiiviseksi tai pysäytetyksi. Viimeisenä määritetään varoituksen sisältö ja tyyppi. Tyyppi voi olla:

- critical
- error
- warning
- information.

Tyyppiä valitessa kannattaa muistaa, kenelle vastaanottajista on sallittu kukin varoitustyyppi. Varoituksen sisällössä voi määritellä varoituksen otsikon, viestin ja numeron, viestien lukumäärän ja viestin saapumisajankohdan sekä liittää viestin XML-muodossa. Tyypeillä ei ole itsessään mitään oletusarvoista määritystä vaan käyttäjä itse määrittelee missä tilanteissa kutakin tyyppiä käytetään. Käytännössä tyypillä siis rajoitetaan, kenelle viestit kulkevat.

3.3.5 Queue Browser

Scribe Consolen Queue Browser (kuva 6) eli jonoselain on MSMQ-jonojen selaamiseen tarkoitettu työkalu.



Kuva 6. Scriben Queue Browser eli jonoselain

Jonoselaimessa käyttäjä voi tarkastella kaikkia jaettuja jonoja sekä tehdä niille erilaisia toimenpiteitä. Ennen kuin jonoselaimen voi ottaa käyttöön, tulee käyttäjän jakaa konsolin Security lehden alta Message Queues välilehden kohdalta haluamansa jonot.

Kun käyttäjä on jakanut jonot, ne tulevat näkyviin jonoselaimeen. Käyttäjä pystyy sitten tarkastelemaan jonojen ominaisuuksia sekä tyhjentämään niitä. Jonojen alta pystyy myös siirtämään, poistamaan ja kopioimaan viestejä. Lisäksi viestien ominaisuuksia ja sisältöä voi tarkastella ja tallentaa.

Jonoselaimen toiminallisuus on kuitenkin melko rajoittunut. Viestien selaaminen on rajattu tuhanteen viestiin ja niistä vain sadalle voi tehdä kerrallaan operaatioita. Jonoselain onkin nimensä mukaan selaustyökalu, josta voi nopeasti tarkastaa jonoissa olevien viestien määrän ja tehdä pienille määrille helposti yksinkertaisia operaatioita. Suuremmille

joukoille tehtävät operaatiot ovatkin parempi tehdä toista kautta, kuten esimerkiksi powershell-skriptillä.

3.3.6 Publishers and Bridges

Konsolista löytyvän Publishers and Bridges -lehden alta käyttäjä voi luoda julkaisijoita sekä siltoja, jotka luovat viestejä Scriben input-jonoon. Julkaisija voi tarkkailla esimerkiksi CRM:ssä tapahtuvia muutoksia tai kyselyn tuloksia ja muodostaa niistä XML-viestejä. Lisäksi XML-viestejä on mahdollista muodostaa sähköpostisillan avulla. Tässä työssä käytettiin Query Publisher -tyyppisiä eli kyselyn tuloksista viestejä muodostavia julkaisijoita. Konsolin kautta käyttäjä voi lisätä, poistaa, aktivoida, deaktivoida, ajaa sekä pysäyttää julkaisijoita. Julkaisijoiden asetuksia on myös mahdollista tarkastella ja muokata.

Julkaisijan luominen koostuu viidestä vaiheesta. Ensin määritellään julkaisijan tyyppi ja nimi. Eri tyyppien lukumäärä riippuu asennetuista adaptereista. Perustyyppeihin kuuluu Query Publisher eli kyselyn julkaisija ja Email Bridge eli sähköpostisilta. Lisäksi Talentumilla on CRM-adapteri, joten valikosta löytyy myös Microsoft Dynamics CRM Publisher, jonka avulla voidaan tarkkailla ja julkaista muutoksia CRM:ssä. Toisessa vaiheessa määritellään yhteys lähteeseen. Esimerkiksi kyselyn julkaisijassa lähde voi olla mikä tahansa tietokanta. Yhteyden muodostamisen jälkeen täytyy määritellä kysely, jonka tulokset julkaistaan. Seuraavaksi määritellään muodostettavien viestien asetukset. Asetuksiin kuuluu juuren nimi, viestin nimi, enkoodaus, prioriteetti ja Scribe-nimike. Lisäksi julkaisijan tuloksia voidaan ryhmitellä sekä vertailla aiempaan tulokseen, ottaen huomioon vain muutokset. Viimeiseksi määritellään, kuinka usein julkaisija ajetaan.

Kaikkia yhteyksiä tehdessä Scribellä tulee ottaa huomioon käyttöoikeudet tietokantoihin. Yhteyden määrittäminen epäonnistuu, jos käyttäjän määrittämillä tunnuksilla ei ole lukuoikeuksia tietokantaan. Julkaisijat käyttävät Scriben palveluja yhdistäessään tietokantoihin. Tästä syystä myös tunnuksella, jolla Scriben palvelut pyörivät, tulee olla lukuoikeudet kyseisiin tietokantoihin.

3.3.7 Administration

Administration-lehden alta käyttäjä pääsee katselemaan ajojen sekä virheilmoitusten lokeja. Lisäksi käyttäjä pääsee käsiksi Site Settings-, Alert Recipients- ja Security-osioihin. Site settings -osioista löytyy Scriben yleisiä asetuksia, kuten koosteiden oletuskansio,

oletusjonot ja sähköpostiasetukset. Alert Recipients -osiosta käyttäjä pystyy määrittelemään kenelle varoituksia lähetetään. Security-osiosta käyttäjä voi asettaa jaetut kansiot, palvelut ja jonot.

3.4 Scriben sisäiset tietokannat

Scribeinternal on Scriben sisäinen SQL Server -tietokanta. Se sisältää ajojen suoritus- ja varoitus-lokit, käyttätietoja, sekä tietoja integraatioprosesseista, monitoreista ja tapahtumista kuten viimeisin ajoaika ja DTS-tiedoston sijainti. Lisäksi kanta sisältää tilastoja Scriben käytöstä.

Scriben kannan lokit eivät tyhjene automaattisesti ja näin saattavat ajan myötä täyttyä, mikä hidastaa Scriben toimivuutta. Tätä varten Scriben kansiossa on olemassa valmis SQL-skripti nimeltä ScribeMaintenance.sql. Skriptin avulla käyttäjä voi puhdistaa kaikki lokitaulut sekä päivittää tietokannan tilastot. Oletuksena skripti poistaa kaikki lokitiedot, jotka ovat yli 365 päivää vanhoja. Skriptin voi ajastaa esimerkiksi käyttämällä Scriben ajastettua integraatioprosessia tai mitä tahansa SQL-ajastajaa.

Tietokannassa on neljä taulua, joihin Scribe kirjoittaa tietoa integraatioprosessien tai DTS-tiedostojen ajoista: executionlog, execlogdetails, transactionerrors ja execlogdetailerrors. Executionlog- ja execlogdetails-taulu sisältävät yleistä tietoa ajetusta prosessista, kuten aloitusaika, kesto, operaatioiden määrä, virheiden määrä, DTS-tiedoston sijainti ja lähteen sekä kohteen sijainti. Transactionerrors- ja execlogdetailerrors-taulu sisältävät tietoa ajojen aikana tapahtuneista virheistä, kuten virheen aiheuttaneen rivin numero ja virheen numerokoodi sekä syy. Scribe kirjoittaa jokaisesta integraatioprosessin ajamisesta yhden rivin executionlog-tauluun sekä execlogdetails-tauluun. Jokaisesta virheestä tulee rivi transactionerrors-tauluun sekä execlogdetailerrors-tauluun.

Scribellä on myös toinen sisäinen tietokanta nimeltä Scribesample. Tätä kantaa ei käytetä missään Scriben sisäisissä toiminnoissa. Se sisältää ainoastaan esimerkkidataa ja on suunniteltu testausta sekä Scriben käytön opettelua varten.

4 Scriben jonopohjaiset integraatioprosessit

Tässä luvussa perehdytään Scriben jonopohjaisten integraatioprosessien tuomiin mahdollisuuksiin sekä tutustutaan viestintämenetelmään, jota ne käyttävät. Lisäksi tarkastellaan, miten jonojen käyttö vaikuttaa virhetilanteista toipumiseen.

Scriben jonopohjaiset integraatioprosessit mahdollistavat usean viestintäprosessorin käytön samanaikaisesti ja näin mahdollistavat ajojen säikeistyksen. Talentumin nykyinen Scribe-lisenssi sisältää kahdeksan viestintäprosessoria, joista seitsemän on tarkoitettu vain jonopohjaisille prosesseille ja yksi kaikille muille. Jonopohjaisilla prosesseilla voidaan siis tehdä seitsemän säikeen ajoja.

Jonopohjaiset integraatioprosessit perustuvat jonoissa liikkuviin XML-viesteihin. Scriben kyselyn julkaisija muodostaa XML-viestit ja lähettää ne MSMQ:n avulla jonoihin, josta jonopohjainen integraatioprosessi saa ne kiinni. Jokainen viestintäprosessori pystyy käsittelemään yhden viestin kerrallaan.

Säikeistyksen lisäksi jonopohjaisilla integraatioprosesseilla on myös muita ominaisuuksia, jotka tekevät niistä paremman vaihtoehdon suurien datamäärien integraatioita tehtäessä. Yksi näistä ominaisuuksista on mahdollisuus laittaa prosessi tauolle ja myöhemmin jatkaa siitä, mihin jäätiin. Muut prosessit voidaan vain pysäyttää kokonaan ja aloittaa alusta, mutta jonopohjaisen prosessin pysähtyessä käsittelemättömät viestit jäävät input-jonoon odottamaan, kunnes niitä kutsutaan. Tämä mahdollistaa jatkamisen siitä, mihin jäätiin, joka on todella tärkeää varsinkin tuotantoympäristöä käytettäessä, jolloin integraatio saattaa hidastaa muuta tekemistä ja pitää väliaikaisesti laittaa tauolle.

Toinen jonojen mahdollistama ominaisuus on epäonnistuneiden rivien automaattinen uudelleenyrittäminen. Tällä saadaan automaattisesti ajettua esimerkiksi aikakatkaisun takia epäonnistuneet viestit uudelleen. Tietenkään uudelleen yrittäminen ei aina auta, jos viestissä on jotain vikaa, jolloin vika pitää manuaalisesti korjata. Tämä kuitenkin vähentää epäonnistumisten määrää suurella aikavälillä.

Scribellä on jo toteutettu integraatioprosesseja, jotka vievät Tabusta dataa CRM:ään. Nämä prosessit ovat ajastettuja prosesseja, jotka toimivat ilman säikeistystä. Prosessien virheet tallentuvat scribe_logs-nimiseen tietokantaan Rejected Source Rows -logiikan avulla. Virheen sattuessa sen aiheuttanut tietue lisätään scribe_logs kannassa olevaan

tauluun. Jokaiselle ajolle on oma virheloki-taulu. Näiden prosessien suorituskyky todettiin liian heikoksi, jonka takia jonopohjaisten integraatioprosessien käyttöönotto on tarpeen. Lisäksi niiden luotettavuus ei ollut tarpeenmukainen. Miljoonia rivejä ajettaessa esiintyi Communication link failure-virhettä, joka viittaa yhteyden katkeamiseen tietokantaan. Tämä aiheutti kuolettavan virheen, jonka seurauksena ajo pysähtyi ja se jouduttiin aloittamaan kokonaan alusta. Jonojen avulla pyritään pääsemään eroon tällaisista tapauksista ja parantamaan ajojen luotettavuutta. Jonon säikeet prosessoivat vain yhden viestin kerrallaan. Kun viesti saa kuolettavan virheen, se siirtyy retry-jonoon ja muiden viestien prosessointi jatkuu normaalisti. Tällöin koko ajoa ei tarvitse aloittaa alusta.

Tarkoituksena on käyttää olemassa olevien prosessien DTS-tiedostoja hyväksi jonopohjaisissa integraatioprosesseissa. DTS-tiedostoja ei siis tarvitse alusta alkaen tehdä uudelleen, vaan olemassa oleviin tehdään tarvittavat muutokset, jotta niitä voidaan käyttää jonojen kanssa.

4.1 Jonoihin perustuva viestintämenetelmä (MSMQ)

MSMQ eli Microsoft Message Queuing on Microsoftin kehittämä jonoihin perustuva viestintämenetelmä, joka mahdollistaa prosessien välisen kommunikaation. MSMQ:n avulla Scriben julkaisijan muodostamat XML-viestit liikkuvat jonojen välillä. Ennen kuin jonopohjaisia integraatioprosesseja voi käyttää, tulee MSMQ olla asennettuna ja konfiguroituna Scriben käyttöön.

4.2 Jonot

Jonot ovat MSMQ:n käyttämiä säilytyspaikkoja viesteille. Viestien lähettäminen ja vastaanottaminen jonojen välillä sekä viestien prosessointi ovat kaikki erillisiä toimintoja toisistaan. Viestit liikkuvat jonojen välillä riippumatta siitä onko kohdesovellus valmis ottamaan ne vastaan (MSMQ Documentation). Scribe vaatii kolme jonoa jonopohjaisten integraatioiden käyttöön. Nämä ovat input-, retry- ja deadmessage-jonot.

Jonot toimivat FIFO-periaatteella (engl. first in first out). Tämä tarkoittaa, että ensimmäinen jonoon tullut viesti lähtee ensimmäisenä liikkeelle, kun integraatioprosessi kutsuu sen tyyppistä viestiä. Tämä pätee kuitenkin vain ryhmitellysti viestityypin ja prioriteetin

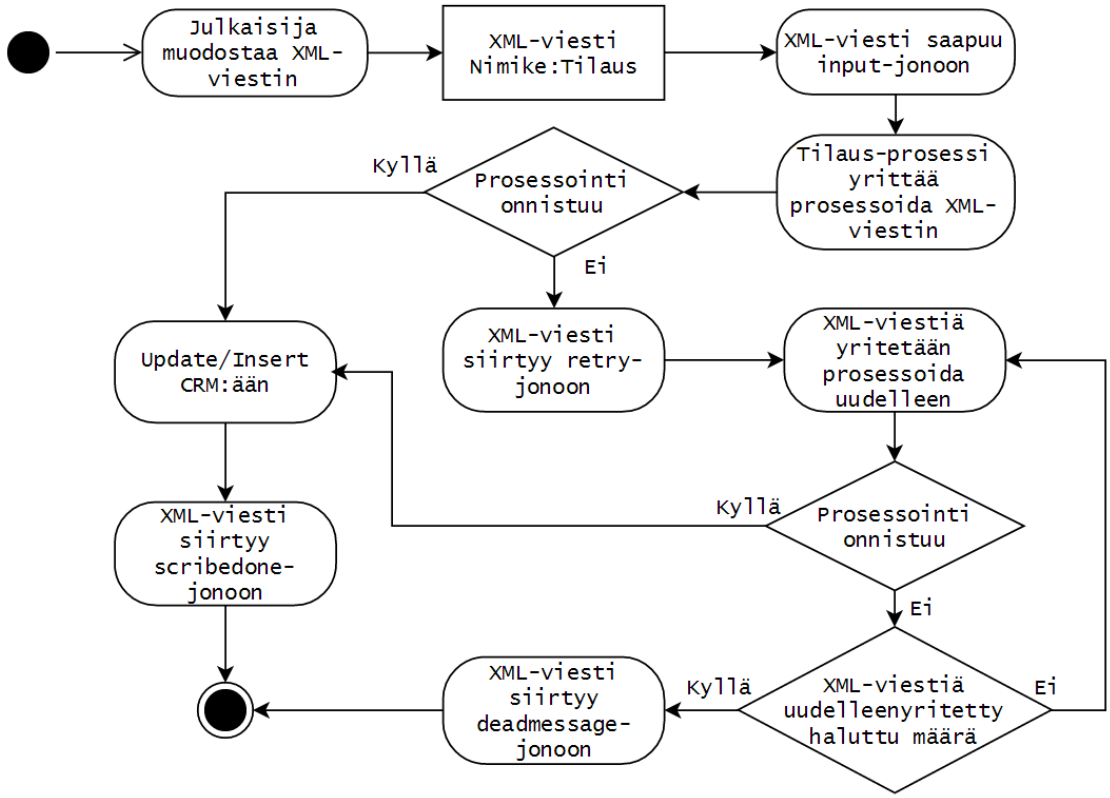
mukaan. Useampaa eri viestityyppiä samaan aikaan kutsuttaessa ensimmäisenä lähtee siis suurimman prioriteetin viestityypin viestit FIFO-periaatteella.

Kaikki julkaisijoiden muodostamat viestit saapuvat ensin input-jonoon. Täällä ne odottavat, kun Scribe tarkistaa, onko kyseisille viesteille vastaavia integraatioprosesseja olemassa vertailemalla viestien ja niiden juurten nimiä sekä mahdollisesti skeemaa. Jos viestille löytyy vastaava prosessi, se siirtyy prosessoitavaksi. Tässä vaiheessa se on kuitenkin edelleen input-jonossa. Viestin prosessoinnin onnistuessa se siirtyy prosessin määrittelemään jonoon. Jos vastaavaa prosessia ei taas löytynyt, viesti päätyy deadmessage-jonoon.

Retry-jonoa käytetään epäonnistuneiden viestien uudelleenprosessointiin. Jokaiselle integraatioprosessille voidaan määritellä, kuinka monta kertaa viestiä yritetään prosessoida uudelleen ennen kuin se siirtyy deadmessage-jonoon. Oletusarvona viestejä yritetään prosessoida uudelleen 120 kertaa 5 sekunnin välein eli yhteensä 10 minuutin ajan. Tämä kuitenkin käyttää viestintäprosessoreja ja näin hidastaa muiden viestien prosessointia.

Kolmen pakollisen jonon lisäksi tässä työssä käytetään neljää muuta jonoa. ScribeDone- ja ScribeDailyDone-nimiset jonot ovat onnistuneita viestejä varten. Lisäksi kahta jonoa ScribeTemp1 ja ScribeTemp2 käytettään väliaikaisina varastoina viesteille.

Kuvasta 7 nähdään, kuinka XML-viestit ja niiden sisältämä data liikkuu Tabusta jonojen kautta CRM:ään. Esimerkkinä tarkastellaan tilauksen tietoja sisältävää XML-viestiä.



Kuva 7. Tilaus-viestin kulku jonoissa

Julkaisijan muodostettua XML-viestin se siirtyy input-jonoon odottamaan prosessointia. Kun tilaus-prosessi havaitsee viestin, prosessi käynnistyy ja yrittää prosessoida kyseisen viestin. Prosessin onnistuessa viestin sisältö viedään CRM:ään Update/Insert-operaation avulla ja XML-viesti siirtyy scribedone-jonoon. Jos viesti epäonnistuu, se siirtyy retry-jonoon. Tilaus-prosessia perustettaessa määriteltiin uudelleenyritysten määrä viiteen kertaan. Retry-jonoon siirtynyttä viestiä yritetään siis prosessoida viisi kertaa uudelleen. Jos uudelleenyritys onnistuu, tapahtuu Update/Insert-operaatio CRM:ään ja XML-viesti siirtyy scribedone-jonoon. Jos kaikki viisi uudelleenyritystä epäonnistuu, viesti siirtyy deadmessage-jonoon. Lopputuloksena onnistuneet viestit ovat Tilaus-nimikkeellä scribedone-jonossa ja epäonnistuneet myös Tilaus-nimikkeellä deadmessage-jonossa.

4.3 Virheiden hallinta ja lokitus

Yksi jonopohjaisten integraatioprosessien keskeisimmistä ominaisuuksista on epäonnistuneiden viestien uudelleenyrittäminen. Viestin prosessoinnin epäonnistuessa se siirtyy retry-jonoon, jossa sitä yritetään prosessoida uudelleen käyttäjän määrittelemien parametrien mukaisesti. Tätä ominaisuutta ei kuitenkaan suositella suoraan käytettäväksi

Rejected Source Rows-logiikan kanssa. Jokaisesta uudelleenyrityksestä syntyisi uusi tietue Rejected Source Rows-tauluun. Tämä tarkoittaisi käytännössä, että viidestä uudelleenyrityksestä tulisi sama rivi tauluun.

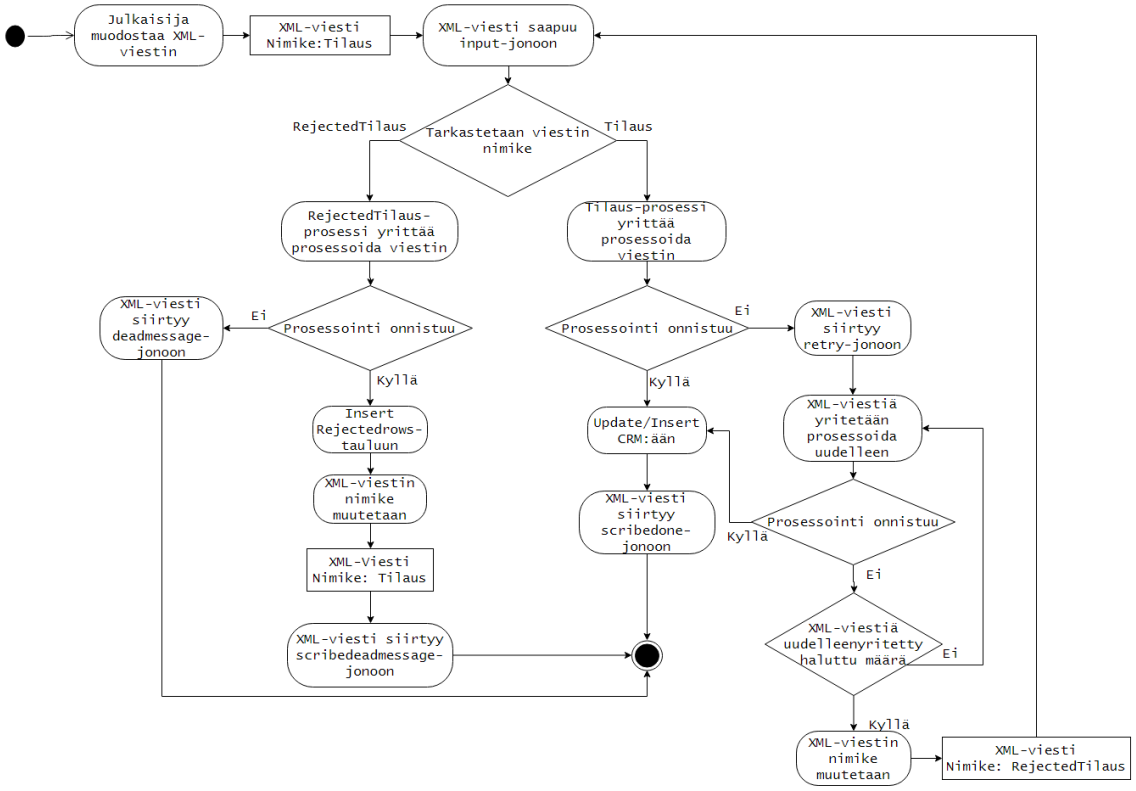
Virheet lokittuvat kuitenkin Scriben sisäiseen tietokantaan execlogdetailerrors-tauluun. Hyvä puoli tässä on, että se tapahtuu automaattisesti eikä se vaadi mitään toimenpiteitä käyttäjältä. Taulusta voi hakea viestin sisällön ja virheilmoituksen SQL-kyselyllä (esimerkkikoodi 1). Se on kuitenkin todella hidasta varsinkin,R jos Scribe on käytössä samaan aikaan tai jos virheitä on suuri määrä.

```
Select distinct
          cast(el.message as xml).value('(/Asiakas/Asiakasnro)[1]',
          'varchar(max)') as asiakasnro
         ,eld.errormessage
from scribe.userview_executionlog el
          inner join scribe.execlogdetailerrors eld
             on el.execid = eld.execid
where collaboration = 'TABU-CRM'
  and name = 'Asiakas Update'
  and fatalerrorcode = 15;
```

Koodiesimerkki 1. virheellisten viestien asiakasnumeroiden haku scribeinternal-kannasta

Rejected Source Rows-logiikan voi kuitenkin ottaa käyttöön, jos halutaan nopeasti tarkastella dataa, joka aiheutti virheen. Tämä vaatii kuitenkin toisen DTS-tiedoston ja integraatioprosessin luomisen. Rejected Source Rows-logiikan avulla data saadaan kuitenkin paljon nopeammin näkyviin vaikka sitä olisikin suuri määrä. Tämä tekee suurten virhemäärien tarkastelusta helpompaa.

Kuvasta 8 nähdään, kuinka XML-viestit ja niiden sisältämä data liikkuu Tabusta CRM:ään, kun Rejected Source Rows-toiminallisuus otetaan käyttöön. Logiikka on hieman monimutkaisempi verrattuna kuvassa 7 nähtyyn logiikkaan ilman Rejected Source Rows -toiminallisuutta.



Kuva 8. Rejected Source Rows -toiminallisuuden ja jonojen aktiviteettikaavio

Logiikka on onnistuneen prosessoinnin tai uudelleenyritysten tapauksessa sama kuin kuvassa 7. Eroavaisuudet tulevat esille, kun viestin prosessointi ei onnistunut edes uudelleenyrittämisen jälkeen. Tässä vaiheessa XML-viestin nimike muutetaan muotoon RejectedTilaus ja se siirretään takaisin input-jonoon. RejectedTilaus-prosessi huomaa XML-viestin jonossa ja yrittää viedä sen Rejected Source Rows-tauluun. Jos tämä ei onnistu, viesti siirtyy deadmessage-jonoon sellaisenaan. Prosessoinnin onnistuessa XML-viestin sisältö lisätään Rejected Source Rows -tauluun ja sen nimike muutetaan takaisin muotoon Tilaus. Tämän jälkeen viesti siirtyy deadmessage-jonoon. Lopputuloksena onnistuneet viestit ovat Tilaus-nimikkeellä scribedone-jonossa. Epäonnistuneet viestit ovat Tilaus-nimikkeellä deadmessage-jonossa. Lisäksi XML-viestit, joiden prosessointi epäonnistui sekä CRM:ään että Rejected Source Rows-tauluun vietäessä, ovat RejectedTilaus-nimikkeellä deadmessage-jonossa. Näin saadaan eroteltua eri tapaukset nimikkeen perusteella.

Kun kumpaakin tapaa käsitellä virheitä testattiin, tultiin siihen lopputulokseen, että Rejected Source Rows -ominaisuuden käyttöönotto ei ole pakollista ja ilman sitä pärjätään hyvin. Virheiden tiedot voidaan hakea suoraan scribeinternal-kannasta. Tällöin päästään

eroon ylimääräisistä prosesseista ja säästetään resursseja onnistuneiden viestien prosessointiin. Tämä helpottaa myös prosessien logiikan muuttamista sekä tekee niistä helpommin ylläpidettäviä.

5 Käyttöönotto ja konfigurointi

Jonopohjaisten integraatioprosessien käyttöönotto koostuu useasta vaiheesta. Ensimmäiseksi MSMQ täytyi asentaa ja konfiguroida Scriben käyttöön. Tämän jälkeen perustettiin kyselyn julkaisija ja muodostetaan XML-esimerkkiviesti, jota käytetään DTS-tiedostoa konfiguroitaessa. Seuraavaksi olemassa olevaa DTS-tiedostoa täytyi muokata niin, että se toimii jonopohjaisten prosessien kanssa. Lopuksi perustettiin jonopohjainen integraatioprosessi Scriben konsolin avulla ja konfiguroitiin se lukemaan julkaisijan muodostamia viestejä. Näiden vaiheiden jälkeen jonopohjainen prosessi oli valmis käytettäväksi.

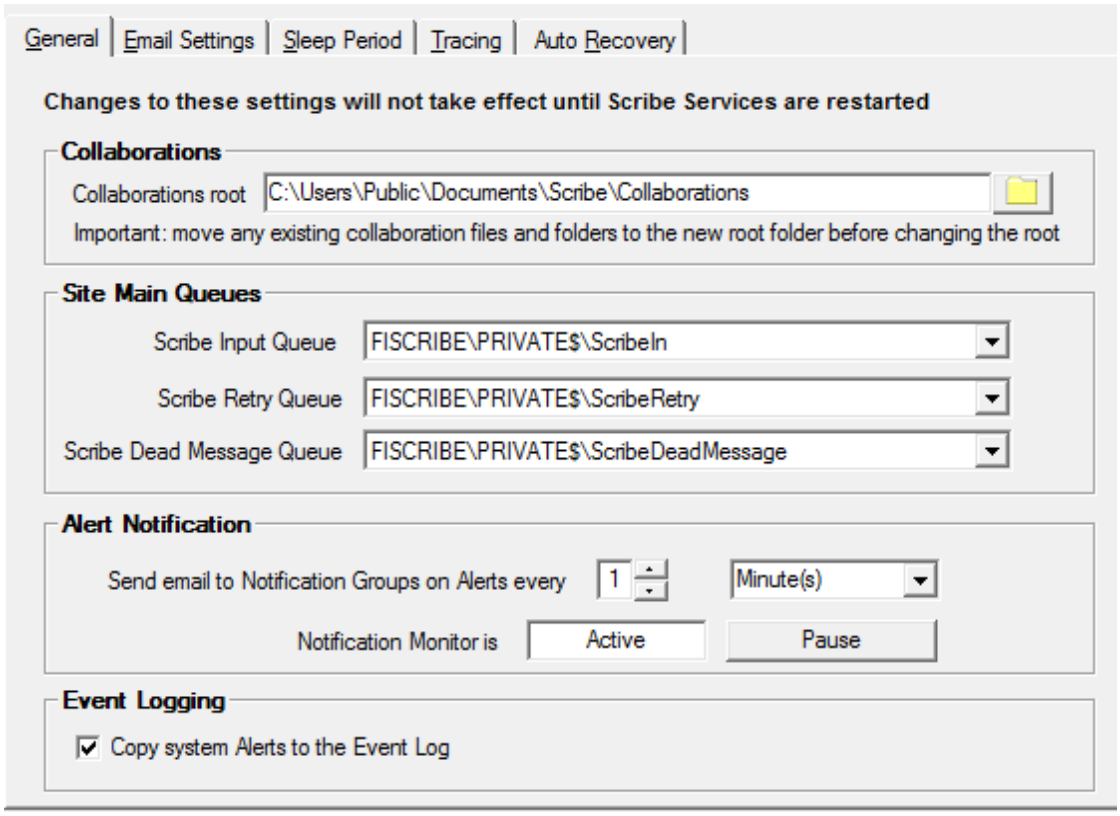
5.1 MSMQ:n asennus

Scribe on asennettuna Windows Server 2012 R2 -palvelimelle nimeltä Fiscribe. Windows Server 2012 R2 mukana tulee MSMQ:n versio 6.3, joka pitää manuaalisesti asentaa. Asennus tapahtuu Server Managerin kautta Add roles and features -velhon avulla. Asennuksen tyypiksi valittiin Role-based or feature based installation. Tämän jälkeen valittiin Fiscribe palvelimeksi, johon asennus tehdään. MSMQ on Features-kohdan alla, joten Server Role -kohdassa ei valittu mitään. Features-kohdasta Message Queuing alta löytyy Message Queuing Services. Tämän alta valittiin asennettavaksi Message Queuing Serverin. Muita palveluja ei tarvinnut asentaa.

MSMQ:n asennuksen jälkeen lisättiin tarvittavat jonot Computer management -työkalun avulla. Se on Scriben konsolin tavoin Microsoft management -konsoliin perustuva sovellus. Services and Applications-lehden alta löytyy Message Queuing -lehti. Tämän alta voidaan lisätä jonoja Private Queues eli yksityisiin jonoihin. Kaikki seitsemän tarvittavaa jonoa lisättiin tänne: scribein, scriberetry, scribedeadmessage, scribedone, scribedailydone, scribetemp1 ja scribetemp2. Lisäksi message queuing -asetuksista lisättiin jonojen

käytössä olevan tallennustilan määrää 20 gigatavuun, jotta jonoissa voi kulkea suuria määriä viestejä.

Jonojen luonnin jälkeen ne piti lisätä Scriben käyttöön Scribe-konsolin avulla. Security valikosta Message Queues -välilehden alta kaikki muodostetut jonot lisättiin jaettuihin jonoihin. Tämän jälkeen Site settings-valikon General-välilehdeltä asetettiin kolme pääjonoa kuvan 9 mukaisesti.

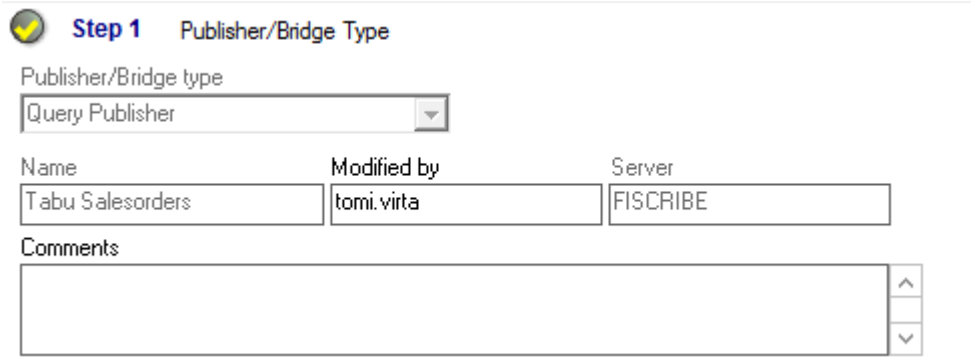


Kuva 9. Pääjonojen asetus

Jonot olivat nyt valmiita Scriben käyttöön.

5.2 Kyselyn julkaisin perustaminen

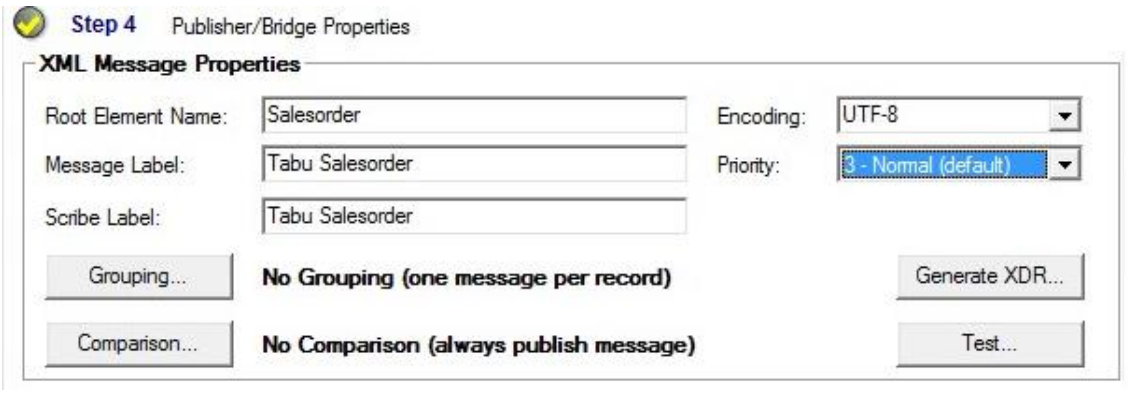
MSMQ:n asennuksen jälkeen täytyi seuraavaksi perustaa julkaisija, jotta saatiin XML-testitiedosto DTS:n konfigurointia varten. Ensimmäisessä vaiheessa asetettiin julkaisijan tyypiksi kyselyn julkaisija ja nimeksi ”Tabu Salesorders” (kuva 10).



Kuva 10. Julkaisijan tyypin ja nimen asettaminen

Toisessa vaiheessa muodostettiin yhteys Tabun tietokantaan. Oracle-adapterilla oli jo perustettu yhteys Tabuun, jota hyödynnettiin tässä. Kolmannessa vaiheessa määriteltiin kysely, jolla XML-viestin sisältö haetaan Tabusta. Tämä kysely saatiin suoraan nykyisten integraatioprosessien DTS-tiedoston lähdekyselystä.

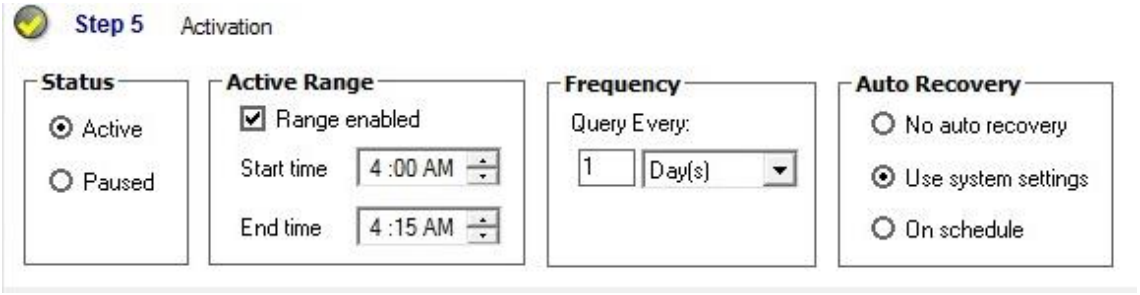
Neljännessä vaiheessa määriteltiin XML-viestiin liittyvät asetukset. Enkoodaus asetettiin UTF-8-muotoon ja juuren nimeksi Salesorder. Prioriteetti jätettiin normaaliksi. Message Label- ja Scribe Label -kenttiin asetettiin arvo ”Tabu Salesorder”. Tämän jälkeen generoitiin XDR-tiedosto ja testi XML-tiedosto. XML-testitiedosto generoitui kuvassa 11 näkyvästä Test-painikkeesta.



Kuva 11. XML-viestin asetusten määrittely

Se muodostui käyttäjän appdata/local/temp/-hakemistoon, josta se pitää käydä kopioimassa haluamaansa kansioon. XDR-tiedosto muodostuu automaattisesti käyttäjän valitsemaan kansioon Generate XDR–painiketta painamalla. Sekä XDR- että XML-tiedostot tallennettiin samaan hakemistoon, johon tuli myös niitä käyttävä DTS-tiedosto.

Viimeisessä vaiheessa asetettiin julkaisija käynnistymään seuraavana yönä klo 4:00 ja 4:15 välisellä ajalla (kuva 12).



Kuva 12. Julkaisijan ajastus

5.3 DTS-tiedoston konfigurointi

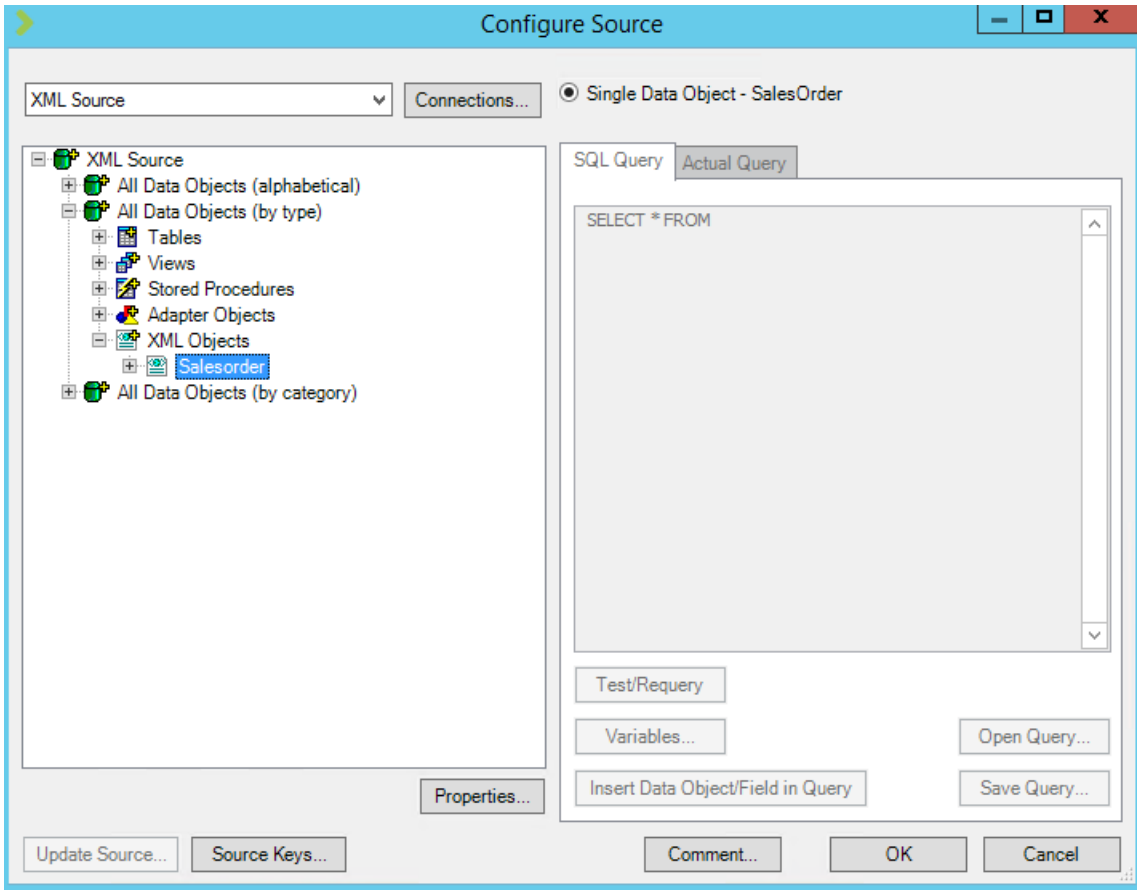
Valmiin DTS-tiedoston voi konfiguroida toimimaan jonopohjaisten integraatioprosessien kanssa muuttamalla lähde XML-viestiksi sekä tekemällä muutama muu tarvittava toimenpide.

Ensimmäisenä muodostettiin yhteys generoituun XML-esimerkkiviestiin. Yhdistäminen tapahtui samasta paikasta kuin muihinkin kohteisiin kuten tietokantoihin yhdistäminen, mutta tyypiksi valittiin XML. Scribe aukaisee tällöin XML-velhon, jonka avulla voi helposti määritellä yhteyden XML-tiedostoon. Ensimmäisessä vaiheessa valittiin, käytetäänkö XML-yhteyttä lähteenä vai kohteena. Tässä tapauksessa XML-viestit ovat lähde, ja CRM on kohde. Seuraavaksi valittiin yhteyden tyyppi kolmesta vaihtoehdosta, jotka olivat

- Dynamic (file or message)
- persistent file
- web /FTP or http.

Näistä valittiin Dynamic eli dynaaminen vaihtoehto, joka mahdollistaa MSMQ-viestien käytön lähteenä. Viimeiseksi määriteltiin XML- ja XDR-tiedostojen sijainti sekä XML-yhteyden nimeksi "XML Source". Tämän jälkeen XML-viesti oli valmis käytettäväksi lähteenä. Seuraavaksi korvattiin nykyinen lähde XML-viestillä Configure Source -valikosta. Lähdeyhteydeksi vaihdettiin juuri perustettu XML Source –niminen yhteys. Scribe kysyi,

halutaanko olemassa olevaa kyselyä käyttää uudessa yhteydessä. Tähän valittiin kyllä, jolloin Scribe säilyttää datamappaukset, vaikka niille ei löytyisi samanimistä vastinetta uudesta lähteestä. Tämän jälkeen valittiin Lähteen tyypiksi Single Data Object eli kaikki kentät valitusta objektista. Objektiksi valittiin XML Objekti, joka näkyy Scribessä sen tiedostonimellä Salesorder (kuva 13).



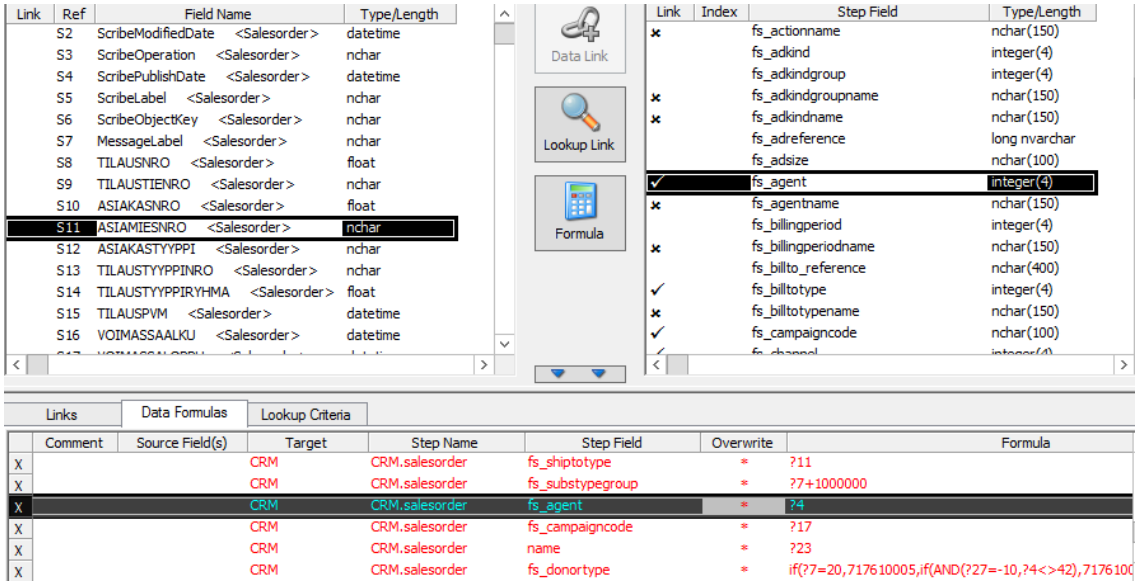
Kuva 13. XML-viestin asettaminen lähteeksi

Tämän jälkeen aukeaa vielä ponnahdusikkuna, jossa voi asettaa toistuvia tietueita. Tätä asetusta käytetään, jos yhdessä XML-viestissä on useampi tietue. Tässä tapauksessa yksi viesti sisältää vain yhden tilauksen, eli ponnahdusikkuna voitiin sulkea muuttamatta mitään asetuksia painamalla OK-painiketta.

Tässä vaiheessa Scribe valitti virheellisistä funktioista, ja jokainen datalinkki muuttuu punaiseksi, joka tarkoittaa manuaalisen korjauksen tarvetta. Tämä johtuu siitä, että jokaisen lähdekentän nimi oli muuttunut hieman erilaiseksi, ja alkuperäisten lähdekenttien

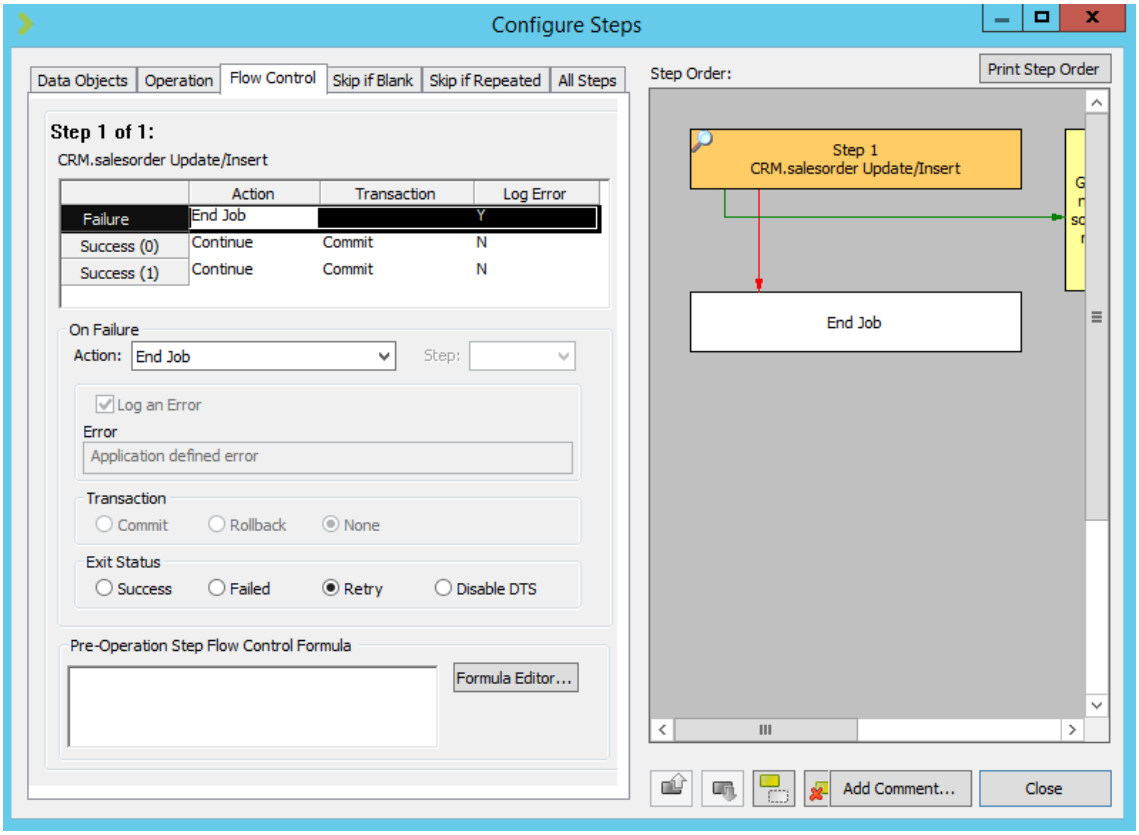
eteen oli tullut seitsemän uutta Scriben XML-viestiin liittyvää kenttää. Esimerkiksi alkuperäinen lähdekenttä S1 Tilausnro löytyi nyt kohdasta S8. Jokaiseen lähdekentän numeroon oli siis lisättävä seitsemän, jotta saatiin tiedon nykyinen sijainti.

Esimerkiksi kuvasta 14 nähdään, kuinka aiemmin S4-kentästä mapattu fs_agent näkyi nyt datakaavoissa punaisena ”?4”-merkintänä. XML-lähteestä tieto löytyi S4+7 eli S11. Datakaavaan piti siis manuaalisesti muokata arvo S11. Tämä sama prosessi täytyi tehdä jokaiselle data- ja lookupkaavalle.



Kuva 14. Workbench-näkymä XML-lähteen asettamisen jälkeen

Kaavojen ja funktioiden korjauksen jälkeen poistettiin Rejected Source Rows -toiminallisuus käytöstä ja poistettiin yhteydet sen ja Tabun tietokantoihin. Tämän jälkeen siirryttiin muokkaamaan kohteen operaatioaskelia. Tässä prosessissa oli vain yksi Update/Insert-askel, joka virhetilanteessa jatkaa seuraavalle riville ja lokittaa virhekoodin scribeinternal-tietokantaan. Flow control-välilehdeltä asetettiin prosessi virhetilanteessa lopettamaan prosessoinnin ja sulkeutumaan Retry-statuksella. Tämä tarkoittaa, että virheen tapahtuessa prosessi lähettää uudelleenyrityskomennon ja viesti siirtyy scriberetry-jonoon (kuva 15).



Kuva 15. Operaatioaskeleen uudelleenyrityksen asetus

Seuraavaksi tarkastettiin All Steps välilehdeltä, että kohdan Step Conditions kaksi asetusta olivat pois päältä. Ensimmäinen asetus ”Allow multi-record matches on updates/deletes” sallisi tässä tapauksessa duplikaattien päivittämisen. Sen ollessa pois päältä saadaan virheilmoitus duplikaateista ja ne voidaan käsin korjata, ennen kuin ajetaan uudestaan. Toinen asetus ”Compare fields on updates” tekee päivityksen vain, jos muutoksia on tapahtunut. Tätä ominaisuutta ei tarvita, koska muutoksia voidaan tarkastella Tabun muutosleiman avulla.

Lopuksi tarkistettiin vielä DTS-asetuksien Performance-välilehdeltä, että ”Turn on % complete” -ominaisuus oli pois päältä. Kun kyseinen ominaisuus on päällä, se hidastaa DTS:n aloitusaikaa. Tästä ominaisuudesta on vain haittaa ajettaessa DTS-tiedostoa konsolin kautta, sillä ainoa hyöty tulee esille ajettaessa suoraan Workbenchiltä. Näiden operaatioiden jälkeen DTS-tiedosto oli valmis käytettäväksi jonopohjaisessa integraatioprosessissa.

5.4 Integraatioprosessin perustaminen

DTS-tiedoston konfiguroimisen jälkeen voitiin perustaa integraatioprosessi, joka käyttää sitä.

Ensimmäiseksi perustettiin uusi kooste tämän projektin integraatioille nimellä TABU – CRM. Seuraavaksi lisättiin integraatioprosessi tämän koosteen alle nimellä Tabu Salesorder Update ja asetettiin se käyttämään konfiguroitua DTS-tiedostoa. Prosessin tyypiksi asetettiin jonopohjainen.

Seuraavassa vaiheessa asetettiin integraatioprosessin viestien asetukset. Näiden asetusten pitää täsmätä julkaisijan asetusten kanssa, jotta integraatioprosessi saa viestit kiinni. Juuren nimeksi tulee siis julkaisijan tavoin Salesorder ja Scribe label- sekä Message label -kenttien arvot ovat Tabu Salesorder. Näillä tiedoilla prosessi huomaa konfiguroidun julkaisijan julkaisemat viestit input-jonossa ja ottaa ne prosessoitavaksi.

Viestien asetusten jälkeen asetettiin viestien siirtymisen säännöt. Viestin prosessoinnin onnistuessa asetettiin viestit siirtymään ScribeDone-jonoon. Uudelleenyritysten lukumääräksi asetettiin viisi. Epäonnistuneiden viestien asetukset jätettiin oletusarvoonsa, eli viesti siirtyvät sellaisenaan scribedeadmessage-jonoon.

Lopuksi asetettiin integraatioprosessi aktiiviseksi. Näin ollen se alkaa prosessoida viestejä heti, kun niitä ilmestyy input jonoon.

5.5 Monitorit

Yksi monitori perustettiin seuraamaan kaikkien päivittäisajojen kulkua. Monitori suorittaa kyselyn CRM-kantaan, joka palauttaa päivitettyjen tietueiden määrän. Ensimmäisessä vaiheessa monitorin tyypiksi asetettiin kysely-monitori ja nimeksi Daily Runs Monitor. Seuraavaksi monitorin lähdeyhteydeksi määriteltiin CRM:n tietokanta. Kolmannessa vaiheessa määriteltiin tietokantakysely ja ehdot, joiden täyttyessä monitori lähettää ilmoituksen sähköpostitse. Monitorin haluttiin lähettävät tieto aina, kun kysely palauttaa rivejä. Ehto oli siis seuraava: jos kyselyn palauttamien rivien lukumäärä ylittää 0 riviä, lähetetään ilmoitus. Ilmoituksen vastaanottajaksi määriteltiin joukko sähköposteja. Tämän jäl-

keen monitori ajastettiin suoritukseen joka aamu klo 6:00. Lopuksi monitori asetettiin aktiiviseksi ja asetettiin monitorin generoiman ilmoituksen asetukset. Ilmoituksen tyypiksi määriteltiin Information, otsikoksi TABU-CRM Daily Runs ja viestiksi Update/Insert Count. Lisäksi viestiin liitettiin kyselyn tulokset.

6 Suorituskykytestit

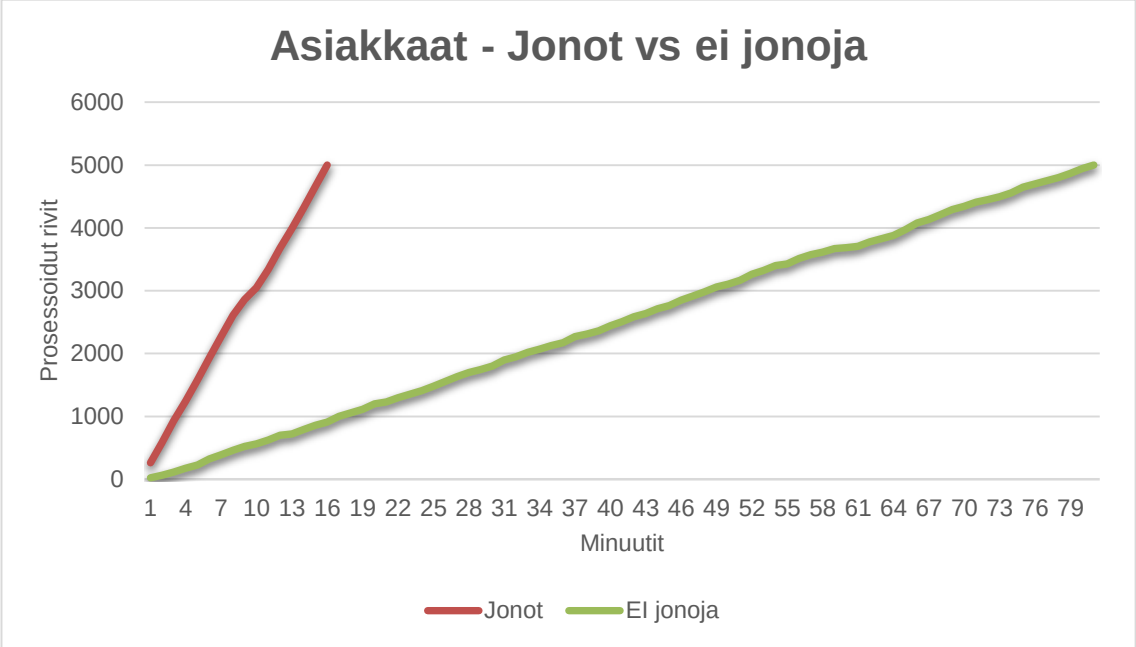
6.1 Testaus

Suorituskykytestien tarkoituksena oli selvittää, kuinka suuri hyöty saavutetaan käyttämällä jonopohjaisia integraatioprosesseja. Tämä tehtiin vertaamalla jonopohjaisten prosessien suorituskykyä säikeettömiin suoraan workbenchiltä ajettuihin prosesseihin. Suorituskyvyn testaamiseen käytettiin kolmea prosessia: Asiakkaiden päivitys, tilausten päivitys ja laskujen päivitys. Ennen suorituskykytestien aloittamista näistä kaikista prosesseista muodostettiin jonopohjaiset integraatioprosessit luvussa 5 kuvattujen käytäntöjen mukaan.

6.1.1 Asiakasajon suorituskyky

Asiakasajo koostuu yhdestä update/insert-operaatioaskeleesta. Se vie Tabusta CRM:ään henkilöasiakkaiden tietoja. Se sisältää 34 data-linkkiä ja yhden lookup-linkin päivitysoperaatiota varten. 16 datalinkkiä sisältävät myös jonkin funktion. Lisäksi ennen jokaista operaatioaskelta tehdään tarkistus, onko asiakas verifioitu CRM:ssä Pre-Operation Step Flow Control Formula –ominaisuuden avulla. Virheen tapahtuessa tietue lisätään Rejected Source Rows -ominaisuuden avulla scribe_logs-tietokannan rejectedasiakas-tauluun.

Suorituskykytestissä ajettiin 5000 asiakasta Tabusta CRM:ään ensin ilman jonoja ja sitten jonojen avulla seitsemällä säikeellä. Ilman jonoja saavutettiin keskimäärin 62 asiakkaan ajo minuutissa. Koko massan ajaminen kesti 81 minuuttia. Minuutissa meni hitaimmillaan 16 asiakasta ja nopeimmillaan 107. Jonojen kanssa saavutettiin keskimäärin 312 asiakkaan prosessointi minuutissa. Koko massa ajettiin noin 16 minuutissa. Hitaimmillaan siirtyi 189 asiakasta minuutissa ja nopeimmillaan 354. Jonoilla saavutettiin siis keskimäärin noin viisinkertainen nopeus (kuva 16).



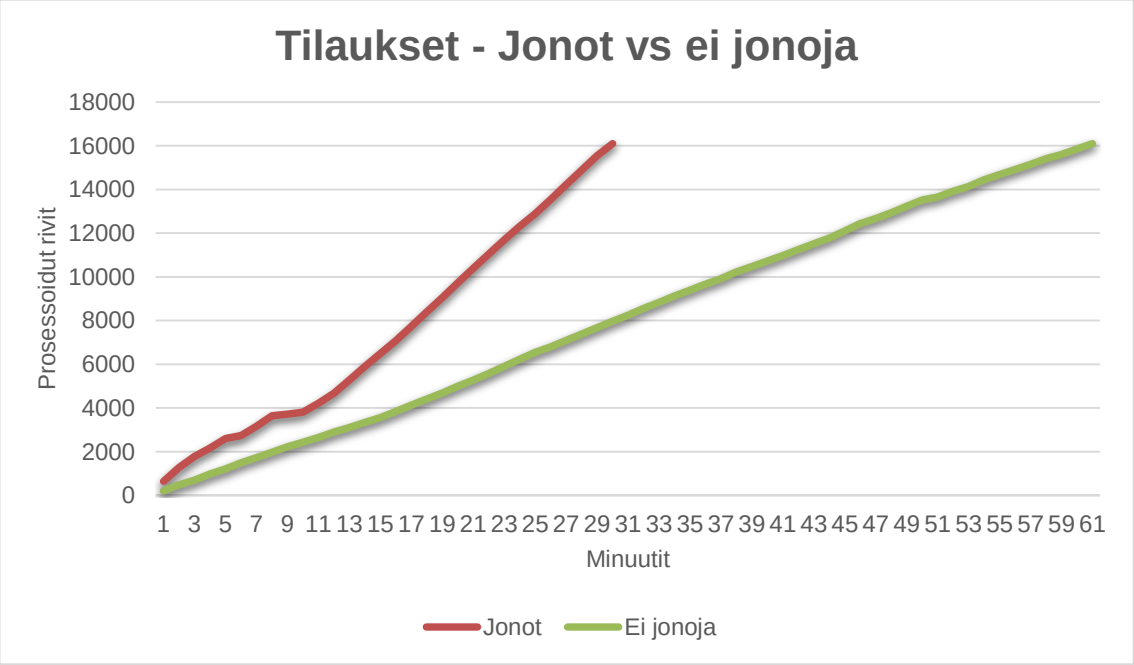
Kuva 16. Asiakasajon suorituskyvyn vertailu

Vaikka säikeitä oli käytössä seitsemän, nopeus oli kuitenkin vain viisinkertainen. Tämä johtuu monesta eri tekijästä. Suurin säikeistettyä prosessointia hidastava tekijä on CRM:n rajapintojen rajoite liittyen samanaikaisiin kutsuihin: CRM rajoittaa samanaikaisten kutsujen määrän kahteen (CRM SDK Documentation 2011). Tämä ei kuitenkaan hidasta Scriben prosessointia. Scribe pystyy edelleen prosessoimaan asiakkaita seitsemän kertaa nopeammin, mutta tietueiden vieminen CRM:ään rajoittuu kahteen säikeeseen. Asiakasajon verifioinnin tarkastuksista sekä data-linkkien funktioista johtuen asiakkaiden prosessointi on todella hidasta. Tämän takia jonojen avulla saavutetaan suurempi kuin kaksinkertainen nopeus.

6.1.2 Tilausajon suorituskyky

Tilausajo koostuu yhdestä update/insert operaatioaskeleesta. Ajo vie Tabusta CRM:ään tilauksia ja niihin liittyviä tietoja. Se sisältää 23 data-linkkiä sekä yhden lookup-linkin. Data-linkeistä 12 sisältää funktion. Virheen tapahtuessa sen aiheuttanut tietue lisätään scribe_logs-tietokantaan rejectedtilaus-tauluun Rejected Source Rows-ominaisuutta hyödyntäen.

Suorituskykytestissä ajettiin 16 095 tilausta Tabusta CRM:ään. Ilman jonoja saavutettiin keskimäärin 264 tilauksen ajo minuutissa. Koko massa ajettiin 61 minuutissa. Hitaimmillaan tilauksia meni 134 minuutissa ja nopeimmillaan 343. Jonojen kanssa seitsemällä säikeellä saavutettiin keskimäärin 537 tilausta minuutissa. Koko massa ajettiin 31 minuutissa. Hitaimmillaan nopeus oli 78 tilausta minuutissa ja nopeimmillaan 680 minuutissa. Jonoilla saavutettiin siis keskimäärin noin kaksinkertainen nopeus. Kuvasta 17 nähdään, kuinka jonojen käyttö vaikutti suorituskykyyn. Jonot siirtävät tilauksia aluksi hieman hitaammin. Tämä johtuu CRM:n samanaikaisesta käytöstä, joka voi aiheuttaa ajon hidastumista. Jonot voisivat siis tuottaa hieman suuremman hyödyn, jos samanaikaista käyttöä ei olisi.

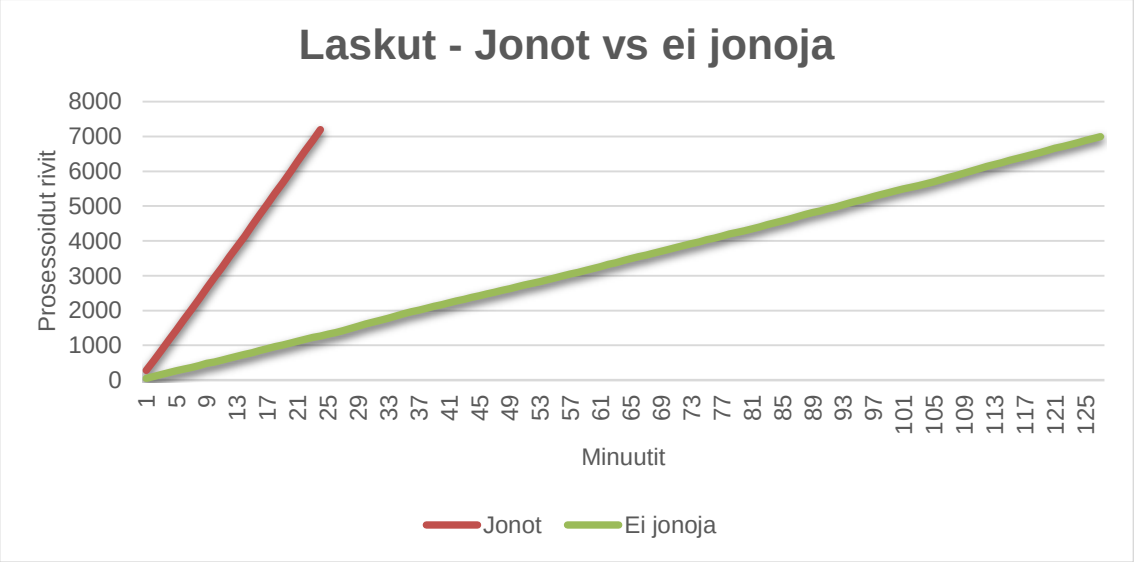


Kuva 17. Tilausajon suorituskyvyn vertailu.

Tällä kertaa saavutettu hyöty oli todella lähellä CRM:n samanaikaisten kutsujen rajaa. Tilausajo on yksinkertaisempi kuin asiakasajo, eikä se vaadi yhtä raskasta prosessointia. Tästä johtuen yhden säikeen ajo on jo melko nopea, eikä seitsemänkertainen prosessointinopeus tuota yhtä suurta hyötyä kuin asiakas-ajossa.

6.1.3 Laskuajon suorituskyky

Laskuajo koostuu yhdestä update/insert-operaatiosta. Se vie Tabusta CRM:ään laskuja sekä niihin liittyvää tietoa. Se sisältää 17 datalinkkiä ja kaksi lookup-linkkiä. Virheet prosessoidaan asiakas- ja tilaus-ajon kaltaisesti Rejected Source Rows -ominaisuuden avulla ja ne lisätään scribe_logs-kannan rejectedlasku-tauluun. CRM:ssä laskut liittyvät aina tilaukseen, joten ennen laskun lisäystä pitää tarkistaa, onko siihen liittyvä tilaus olemassa. Tämä tehdään Pre-Operation Step Flow Control Formula -ominaisuuden avulla. Kuvasta 18 nähdään, kuinka jonojen käyttöönotto vaikutti lasku-ajon suorituskykyyn.



Kuva 18. Laskuajon suorituskyvyn vertailu

Suorituskykytestissä ajettiin 7200 laskua Tabusta CRM:ään. Koko massan ajaminen kesti ilman jonoja 127 minuuttia. Nopeus oli tällöin 55 laskua minuutissa. Nopein saavutettu vauhti oli 66 laskua minuutissa ja hitaimmillaan laskuja siirtyi 44 minuutissa. Jonoja hyödyntäen seitsemällä säikeellä saavutettu keskinopeus oli 300 laskua minuutissa. Koko massan ajamiseen meni tällöin 24 minuuttia. Suurin saavutettu nopeus oli 319 laskua minuutissa. Hitaimmillaan siirtyi 278 laskua minuutissa. Laskujen tapauksessa jonoilla saavutettiin siis noin 5,45-kertainen nopeus.

6.2 Ilmenneet ongelmat

Suorituskykytestejä ajettaessa ilmeni ongelmia, joita ei ennen jonojen käyttöä huomattu. Pienissä ajoissa, kuten näissä suorituskykytesteissä ongelmat olivat todella pieniä. Suurempia massoja ajettaessa näistä pienistä ongelmista koituisi kuitenkin huonoja seurauksia.

6.2.1 Lokien täyttyminen

Scribe muodostaa jokaisesta ajosta yhden rivin scribeinternal-tietokantaan executionlog-tauluun. Jokaisesta virheestä tulee lisäksi rivi transactionerrors-tauluun. Jonopohjaiset integraatioprosessit käyttävät tätä samaa käytäntöä, mutta jokaisen tietueen prosessointi lasketaan omaksi ajokseen. Tästä seuraa, että ajettaessa esimerkiksi miljoona riviä jonopohjaisen integraatioprosessin läpi, muodostuu executionlog-tauluun miljoona riviä. Tämä aiheuttaa executionlog-taulun nopean täyttymisen.

Lokitauluja ei voida kuitenkaan kokonaan tyhjentää esimerkiksi SQL:n Truncate table-komennolla, joka poistaa kaikki rivit määritellystä taulusta. Tämä johtuu lokitaulussa säilytettävästä tiedosta, kuten viimeisen ajon ajankohdasta. Tätä tietoa käytetään ajojen ajastuksessa. Lisäksi lokitaulusta löytyy tietoja tapahtuneista virheistä. Ilman tätä tietoa voi olla vaikeaa selvittää virheiden syitä.

Tämä ongelma on kuitenkin helposti kontrolloitavissa Scriben lokitaulujen Scribemaintenance-nimisen puhdistus-skriptin avulla. Skriptin voi esimerkiksi ajastaa kerran viikossa puhdistamaan osan taulusta. Scribeä ei voi kuitenkaan käyttää samanaikaisesti.

MSMQ säilyttää kaikki viestit jonoissa, kunnes ne poistetaan tai siirretään muualle. Scriben ajot eivät poista viestejä missään vaiheessa. Tämä tarkoittaa sitä, että jonojen putsaus on käyttäjän vastuulla. Powershell-skriptien avulla voidaan helposti automatisoida jonojen putsaus. Esimerkiksi jonon viestien ylittäessä tietyn lukumäärän, voidaan jono putsata purge-komennolla.

Jononen putsaus ei kuitenkaan poista viestejä lopullisesti, vaan siirtää ne MSMQ:n järjestelmäjonoon nimeltä deadletter. Täältä ne voidaan vielä erikseen putsata, joka poistaa ne lopullisesti.

6.2.2 CRM:n järjestelmätöiden täyttyminen

CRM:ssä voidaan automatisoida operaatioita järjestelmätöiden avulla. Talentumilla on CRM:ssä käytössä useita järjestelmätöitä, jotka laukeavat automaattisesti esimerkiksi uusien tilausten tai laskujen muodostuessa. Esimerkiksi tilauksia tai laskuja ajettaessa jonopohjaisten integraatioiden läpi useita satoja minuutissa järjestelmätöitä laukeaa myös satoja. Ne prosessoituvat hitaammin kuin Scribe syöttää dataa, joka johtaa niiden kasaantumiseen. Tämä hidastaa koko CRM:n toimivuutta sekä ajoja. Lisäksi niistä muodostuu rivejä CRM:n asyncoperation-tauluun. Tätä varten on hyvä olla yksi järjestelmätyö, joka puhdistaa tätä taulua automaattisesti, jotta se ei täyty samalla tavalla, kuten Scriben loki-taulu.

Tämä ongelma ratkaistiin ottamalla kaikki järjestelmätyöt pois päältä, jotka laukeavat Scriben ajoissa eivätkä ole kriittisiä CRM:n toiminnan kannalta.

6.2.3 Jonoselaimen rajoitteet

Suorituskykytestien lisäksi suoritettiin testi, jossa testattiin prosessien käyttäytymistä virhetilanteissa. Testissä ajettiin 10 000 riviä väärässä muodossa olevia päivämääriä, joista kaikki epäonnistuivat. Prosessit toimivat oikein siirtäen virheelliset tietueet ensin retry-jonoon ja sitä kautta scribedeadmessage-jonoon. Tässä vaiheessa huomattiin kuitenkin, että jonoselaimella on todella vaikeaa käsitellä suuria määriä jonoissa olevia XML-viestejä.

Selaimella pystyy siirtämään maksimissaan 100 viestiä kerrallaan jonosta toiseen. Jos haluttaisiin siis yrittää vielä uudelleen näitä 10 000 scribedeadmessage-jonossa olevia viestejä, ne pitäisi 100 kerrallaan siirtää scribein-jonoon manuaalisesti. Tämä tarkoittaa siis käytännössä viestien siirtämistä 100 kertaa raahaa ja pudota–menetelmällä. Lisäksi selaimella voi tarkastella vain 1000 ensimmäistä viestiä jonossa.

Viestien siirtoja varten kirjoitettiin powershell-skriptit, jotka pystyvät käsittelemään tuhansia viestejä nopeasti ja vaivattomasti. Seuraava skripti (koodiesimerkki 2) siirtää scribedeadmessage-jonosta löytyvät tilauksiin liittyvät viestit takaisin scribein-jonoon.

```
#Ladataan viestintä DDL
[Reflection.Assembly]::LoadWithPartialName("System.Messaging")
```

```
#Määritellään jonot
$Input = new-object System.Messaging.MessageQueue ".\private\scribein"
$Deadmsg = new-object System.Messaging.MessageQueue
".\private\scribedeadmessage"
$Temp1 = new-object System.Messaging.MessageQueue ".\private\scribetemp1"

$i = 1
do {#Tarkastetaan, onko viestin nimike Tabu Salesorder
   if ($Deadmsg.Peek().Label -eq "Tabu Salesorder"){
          $Input.Send($Deadmsg.Receive()) #Lähetetään viesti input-jonoon
          }
   else {
          $Temp1.send($Deadmsg.Receive()) #Lähetetään viesti temp1-jonoon
          }
   $i++
   }
while ($i -le 600) # 600 = Viestien lkm deadmessage jonossa
```

Koodiesimerkki 2. Tilaus-viestien siirto deadmessage-jonosta input-jonoon

Skriptissä käytetään .NET-ohjelmistokehyksen System.Messaging-nimiavaruuden MessageQueue-luokkaa ja sen kolmea metodia. Peek()-metodi hakee jonon ensimmäisen viestin poistamatta sitä jonosta. Send()-metodi lähettää jonon ensimmäisen viestin sille parametrina annettuun jonoon. Receive()-metodin avulla jono vastaanottaa sille lähetetyn viestin. (System Messaging Namespace Documentation)

Peek()-metodin avulla tarkistetaan, onko ensimmäisen viestin nimike Tabu Salesorder. Nämä viestit liittyvät juuri Tabusta tulleisiin tilauksiin. Viestit siirretään send()- ja receive()-metodien avulla deadmessage-jonosta input-jonoon. Muut viestit siirretään temp1-jonoon. Tämä toistetaan kaikille deadmessage-jonon 600:lle viestille. Skriptillä pystytään siirtämään yli 30 000 viestiä minuutissa.

7 Yhteenveto

Työssä tutustuttiin Talentumin asiakkuudenhallintajärjestelmiin ja Scribe Insight-integraatiotyökaluun sekä sen ominaisuuksiin. Tämän jälkeen perehdyttiin MSMQ:n toimintaan sekä Scriben jonopohjaisiin integraatioprosesseihin. Oletuksena oli, että jonopohjaiset integraatioprosessit parantaisivat integraatioiden suorituskykyä sekä luotettavuutta.

Jonopohjaiset integraatioprosessien käyttöönotto sujui hyvin. Ongelmat ilmenivät vasta suorituskykytesteissä. Jonopohjaiset prosessit generoivat suuremman määrän lokitietoa Scriben kantaan. Lisäksi nopeampi integraatio aiheutti CRM:n järjestelmätöiden kasaan-

tumista. Testien perusteella todettiin, että jonopohjaisten integraatioprosessien suorituskyky oli parempi kuin ilman jonoja toteutettujen prosessien. Parhaimmillaan päästiin jopa yli viisinkertaisiin nopeuksiin. Lisäksi päästiin eroon monista timeout-tyyppisistä virheistä jonojen uudelleenyritys-toiminallisuuden avulla. Ongelmaksi koitui myös Scriben jonoselaimen rajoitteet. Tämän takia jouduttiin kirjoittamaan powershell-skriptejä, jotka siirsivät viestejä jonojen välillä.

Jonopohjaiset integraatioprosessit osoittautuivat todella hyödyllisiksi nopeuden ja luotettavuuden takia. Lopuksi ne ajastettiin käynnistymään automaattisesti joka yö. Työn tavoitteet saavutettiin ja jonopohjaisista integraatioprosessit tulevat olemaan hyödyllisiä myös tulevissa integraatioprojekteissa. Jonoja täytyy kuitenkin myös jatkossa ylläpitää manuaalisesti. Tämä tarkoittaa jonojen siivoilua ja tarkkailua sekä epäonnistuneiden päivittäisajojen ongelmien selvittelyä. Jatkotoimenpiteinä voisi olla jonojen ja lokikantojen puhdistamisen automatisointi esimerkiksi powershell-skriptejä hyödyntäen.

Lähteet

Beckner, Mark. 2015. Using Scribe Insight. Apress.

Scribe Help Library, verkkodokumentti. <http://help.scribesoft.com/scribeinsight/en/>
Luettu 20.2.2016.

System Messaging Namespace Documentation, verkkodokumentti. <https://msdn.microsoft.com/en-us/library/system.messaging(v=vs.110).aspx?tduid=(ce56304b55e1bd40cdfe105f166614a7)(256380)(2459594)(TnL5HPStwNw-zeO5iVOpX154DPmjG.KoGA)()>
Luettu 1.4.2016.

CRM SDK Documentation 2011.

MSMQ Documentation, verkkodokumentti. < https://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx>
Luettu 1.4.2016.