TAMPEREEN
AMMATTIKORKEAKOULU

PLUGINSYNC: TYÖKALU PLUGINIEN AUTOMAATTISEEN REKISTERÖIMISEEN MICROSOFTIN CRM-JÄRJESTELMIIN

Hannu Hytönen

Opinnäytetyö
Huhtikuu 2019
Tieto- ja viestintätekniikka
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

HYTÖNEN, HANNU:
PluginSync: Työkalu pluginien automaattiseen rekisteröimiseen Microsoftin CRM-järjestelmiin

Opinnäytetyö 49 sivua, joista liitteitä 3 sivua
Huhtikuu 2019

Työn tavoitteena oli toteuttaa työkalu, jolla voi ensisijaisesti rekisteröidä sovelluslaajennoksia, eli plugineja, Microsoftin CRM-toiminnanohjausjärjestelmiin. Työkalun tarkoituksena oli sovelluslaajennoksien kehitysprosessin parantaminen. Työ tehtiin Accountor Enterprise Solutions Oy nimiselle yritykselle.

Työssä toteutettiin komentorivisovellus. Sillä pystyy automaattisesti rekisteröimään sovelluslaajennoksia CRM-järjestelmiin. Sovellus toteutettiin niin, että sen suorituksen pystyi skriptaamaan. Sovellukselle toteutettiin myös kaksi toissijaista toimintoa. Ne ovat rekisteröityjen laajennoksien kongifuraatioiden validoiminen sekä rekisteröintiohjeiden luonti rekisteröidyille laajennoksille.

Sovellusta testattiin suotuisissa testiolosuhteissa, jotta sen voitiin todeta toimivan oikein. Sovelluksen kykyä toipua virhetilanteista ei testattu. Kehitysideoita sovellukselle ovat kattavampi testaaminen, virheidensietokyvyn kehittäminen ja erinäisten ominaisuuksien paranteleminen. Toisin sanoen sovelluksen toimintavarmuutta ja käyttömukavuutta täytyy kehittää.

Asiasanat: ohjelmistokehitys, microsoft crm, dynamics 365

ABSTRACT

Tampere University of Applied Sciences
ICT Engineering
Embedded Systems and Electronics

HYTÖNEN, HANNU:
PluginSync: A Tool for Automatic Registration of Plugins into Microsoft CRM systems

Bachelor's thesis 49 pages, appendices 3 pages
April 2019

The goal of this thesis was to implement a tool which can be used to install plugins automatically into Microsoft CRM systems. The purpose of the tool was to improve the plugin development process. The tool was made for a company named Accountor Enterprise Solution Oy.

As a result of this thesis a command line program was developed to enable automatic registration of plugins into CRM systems with it. The program was developed so that its execution could be scripted. There were also two secondary functionalities that were implemented. Those are the validation of already registered plugins and registration info generation for existing plugins.

The program was tested under favorable conditions. The result of testing was that all the implemented functionalities work as intended. However, the program was not tested for error recovery and therefore could be the next step in this application. There is a need for improvement in reliability and usability.

Key words: software development, microsoft crm, dynamics 365

SISÄLLYS

LYHENTEET JA TERMIT

CRM	Customer Relations Management, asiakkuudenhallinta tai MS CRM -toiminnanohjausjärjestelmä.
Plugin	MS CRM:n sovelluslaajennos, jolla lisätään järjestelmään mukautettua liiketoimintalogiikkaa.
Assembly	Laajennuskokoonpano. Käännetty koodikirjasto. Joko suoritettava ohjelma (exe) tai sovelluskirjasto (dll)
Step	Pluginin käsittelyvaihe
Image	Entiteettikuva. Sisältää pluginiin liittyvän entiteetin tietoja
D365	Dynamics 365 -toiminnanohjausjärjestelmä
GAC	Global Assembly Cache. Laitteen laajuinen säiliö laajennuskokoonpanoille
SDK	Software Development Kit. Tietyn ohjelman tai järjestelmän kehitykseen tarkoitettu työkalupaketti.
GUID	Globally Unique Identifier. Tunniste, joka on standardien mukaan luotuna uniikki.
ID	Identifier, tunniste
int	Integer, kokonaisluku
string	Merkkijono

1 JOHDANTO

Tämä opinnäytetyö on tehty Accountor Enterprise Solutions Oy -yritykselle. He halusivat automaattisen työkalun, jolla voi rekisteröidä plugineja Microsoft Dynamics 365 sekä aikaisempiin MS CRM -toiminnanohjausjärjestelmiin.

Ilman automaattista työkalua pluginit rekisteröidään järjestelmään käsin. Pluginien rekisteröinti käsin vie paljon aikaa ja siinä tapahtuu helposti virheitä. Automaattinen rekisteröinti säästäisi aikaa ja ehkäisisi virheellisiä rekisteröintejä. Rekisteröintiohjeet toimivat samalla sekä määrityksinä työkalulle että dokumentointina pluginin rekisteröinnistä. Työkalun käyttöönotto siis helpottaa myös tehtyjen pluginien dokumentointia.

Työn tavoitteena oli toteuttaa konsoliohjelma, joka osaa sille annettujen määrityksien perusteella rekisteröidä plugineja automaattisesti. Lisäksi ohjelman tulee osata validoida D365-organisaatiossa olevien plugien rekisteröinti haluttua määritystä vastaan ja tuottaa rekisteröintiohjeet halutun organisaation rekisteröidyille plugineille. Validointi ja rekisteröintiohjeiden luonti toteutetaan siksi, että vanhoille projekteille tehdyille plugineille saadaan luotua nopeasti ja varmasti määritykset työkalua varten.

Työn pohjana käytetään Microsoftin julkaisemaa MS CRM 2011 Plugin registration tool -ohjelman koodia. Plugin registration tool 2011 on Microsoftin kehittämä sovellus, jolla voi rekisteröidä käsin plugineja CRM 2011:een ja aikaisempiin versioihin. CRM:stä ja registration toolista on julkaistu useampi uusi versio, mutta Microsoft ei ole julkaissut niiden lähdekoodia.

2 TAUSTA

2.1 Microsoft CRM -toiminnanohjausjärjestelmä

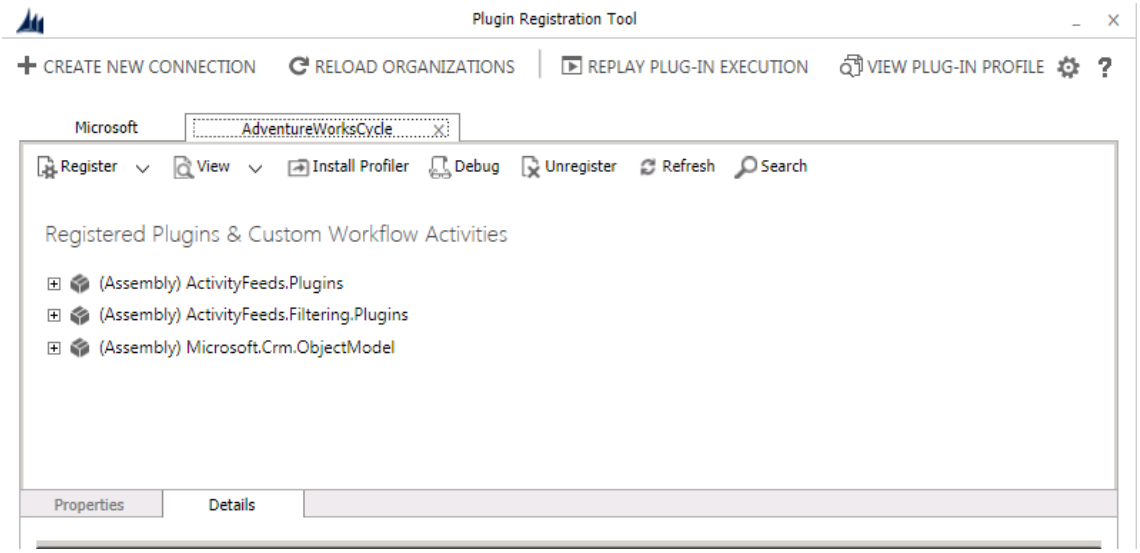
Microsoft Dynamics 365 (D365) on Microsoftin kehittämä asiakkuudenhallinta- ja toiminnanohjausjärjestelmä. Se pohjautuu Microsoftin vanhoihin ohjelmiin, jotka ovat Microsoft Dynamics GP, -NAV, -AX ja -CRM. D365 on järjestelmä, joka on yhdistänyt vanhat erilliset ohjelmat yhdeksi kokonaisuudeksi. Se on tarjolla sekä pilvipalveluna että on-premisena. (Molag 2016). D365 on CRM-tuoteperheen uusin versio. Vanhempia versioita ovat esimerkiksi Dynamics CRM 2016, -2015, -2013, -2011 ja -4.0 (Build Numbers N.d). Accountor Enterprise Solutions tekee kehitystyötä kaikilla edellä mainituilla CRM-versioilla. Tämän takia toteutettavan työkalun tulee tukea kaikkia näitä versioita.

CRM-järjestelmiä voi muokata monin eri tavoin oletustyökaluilla. Esimerkiksi on mahdollista muokata näkymiä, prosesseja, webresursseja, käyttöoikeuksia ja paljon muuta. Oletustyökaluilla tehdyillä mukautuksilla hallinnoidaan yksinkertaisia asioita. Kun tarvitaan esimerkiksi raskasta tai monimutkaista laskentaa, voidaan kehittää CRM:ään lisäosa, eli plugin. (Microsoft 2017a).

2.2 Plugin Registration Tool -työkalu

Plugin Registration Tool on Microsoftin kehittämä työkalu, jolla voidaan rekisteröidä plugineita D365:een. Se on osa CRM SDK:ta, jossa on myös muuta CRM-kehitykseen liittyvää dokumentaatiota, esimerkkikoodia ja työkaluja. (365 Blog 2017).

Plugin Registration Toolilla voi rekisteröidä ja konfiguroida plugineja D365:een. Plugin Registration Tool on graafinen työkalu. Sitä on helppo käyttää, mutta monen pluginin konfigurointi käsin on hidasta. Kuvassa 1 on Plugin Registration Toolin päänäkymä.



KUVA 1. Plugin Registration Toolin päänäkymä (Microsoft 2018)

Päänäkymän yläreunassa on painikkeita, joilla voi luoda uuden yhteyden tai ladata uudestaan olemassa olevan yhteyden. Yläreunasta löytyy myös painikkeet “Replay plug-in execution” ja “View plug-in profile”, jotka ovat pluginien debuggaamiseen tarkoitettuja aputyökaluja. Pluginien debuggaus ei ole tämän työn kannalta oleellista.

Päänäkymässä on listattuna myös organisaatioon rekisteröidyt assemblyt. Assemblyjen vasemmalta olevasta +-painikkeesta voi avata listan, josta näkyy assemblyssä olevat pluginit. Pluginin kohdalta listaa voi laajentaa, jos sille on rekisteröity steppejä. Tällöin näkee tietylle pluginille rekisteröidyt stepit. Edelleen steppien kohdalta listaa voi laajentaa, jos stepille on rekisteröity imageja.

2.3 Sovelluslaajennokset

Sovelluslaajennokset, eli pluginit, ovat D365:een vietävää mukautettua liiketoimintalogiikkaa, jolla muutetaan järjestelmän oletustoimintaa. Pluginin voi ajatella myös olevan D365:ssä tapahtuvien tapahtumien käsittelijä. Pluginit suoritetaan palvelimella, eikä niillä voi toteuttaa mitään, mikä vaatisi käyttöliittymän. Pluginien suoritus tapahtuu joko synkronisesti tai asynkronisesti. (Microsoft 2017a, CRM Book N.d)

Pluginin lisäämistä osaksi D365:ttä kutsutaan pluginin rekisteröinniksi. Rekisteröinnin voi tehdä kahdella tapaa, joko Microsoftin tarjoamalla valmiilla työkalulla Plugin

Registration Toolilla tai ohjelmallisesti. Microsoftin suosittelema tapa on Plugin Registration Tool. (Microsoft 2017c).

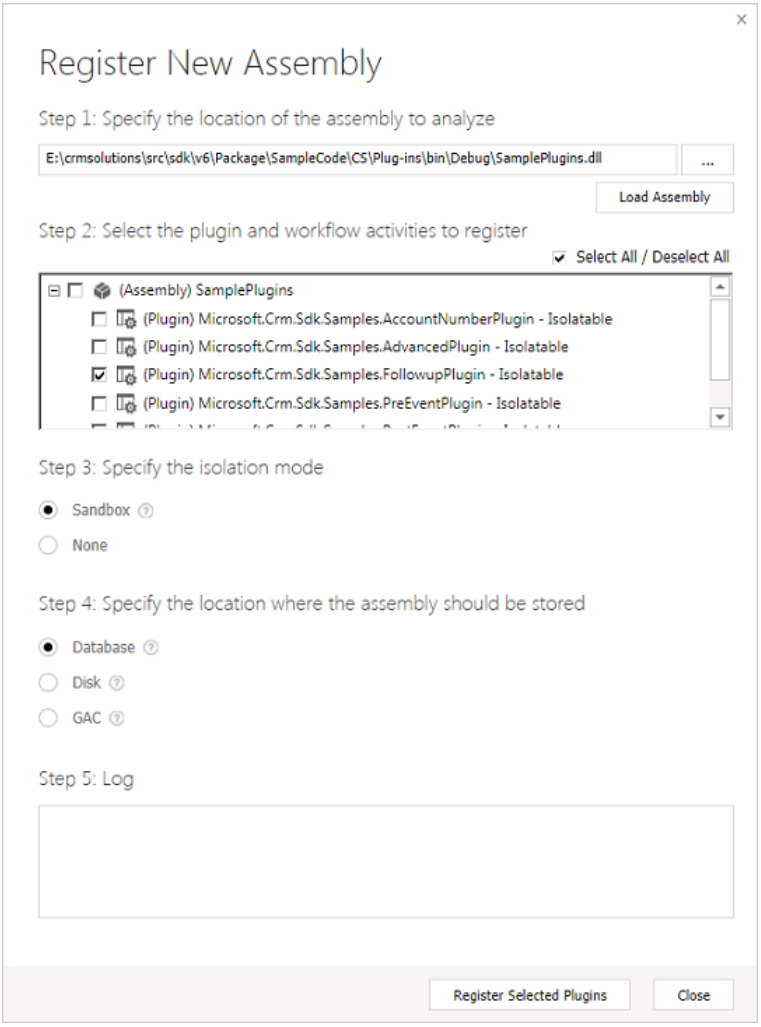
Pluginit ovat yksinkertaisimmillaan luokkia, jotka toteuttavat Microsoftin SDK:n mukaisen rajapinnan IPlugin. Kuvassa 2 on esimerkki luokasta, joka on plugin, mutta joka ei tee mitään. (Microsoft N.d.). Se perii IPlugin-rajapinnan ja se toteuttaa rajapinnan vaatiman Execute-metodin. Metodin sisässä ei ole koodia, joten plugin ei tee mitään. Pluginin rakenne on oikeanlainen, joten sen voisi rekisteröidä CRM:ään.

```
using System;
using System.ServiceModel;
using Microsoft.Xrm.Sdk;

public class MyPlugin: IPlugin
{
    public void Execute(IServiceProvider serviceProvider)
    {

    }
}
```

KUVA 2. Yksinkertainen plugin

Pluginien tarkoitus on mahdollistaa sellaisten toimintojen toteuttaminen, joita ei vakiotyökaluilla voi tehdä. Esimerkkitapauksia ovat muun muassa monimutkainen laskenta, ulkoisien järjestelmien kanssa kommunikoiminen ja uusien tietueiden luominen tietystä herätteestä. CRM:n vakiotyökaluilla ei pysty edellä mainittuja asioita tekemään.

Kun Plugin Registration Tool -työkalulla rekisteröidään assembly, rekisteröitävät pluginit valitaan samassa yhteydessä. Työkalulla valitaan ensin haluttu assembly, jonka jälkeen työkalu lataa assemblyssä olevat pluginit. Käyttöliittymästä valitaan halutut pluginit, jotka halutaan rekisteröidä. Kuvassa 3 on esimerkki pluginien rekisteröinnistä.



KUVA 3. Pluginien rekisteröinti (Microsoft 2018)

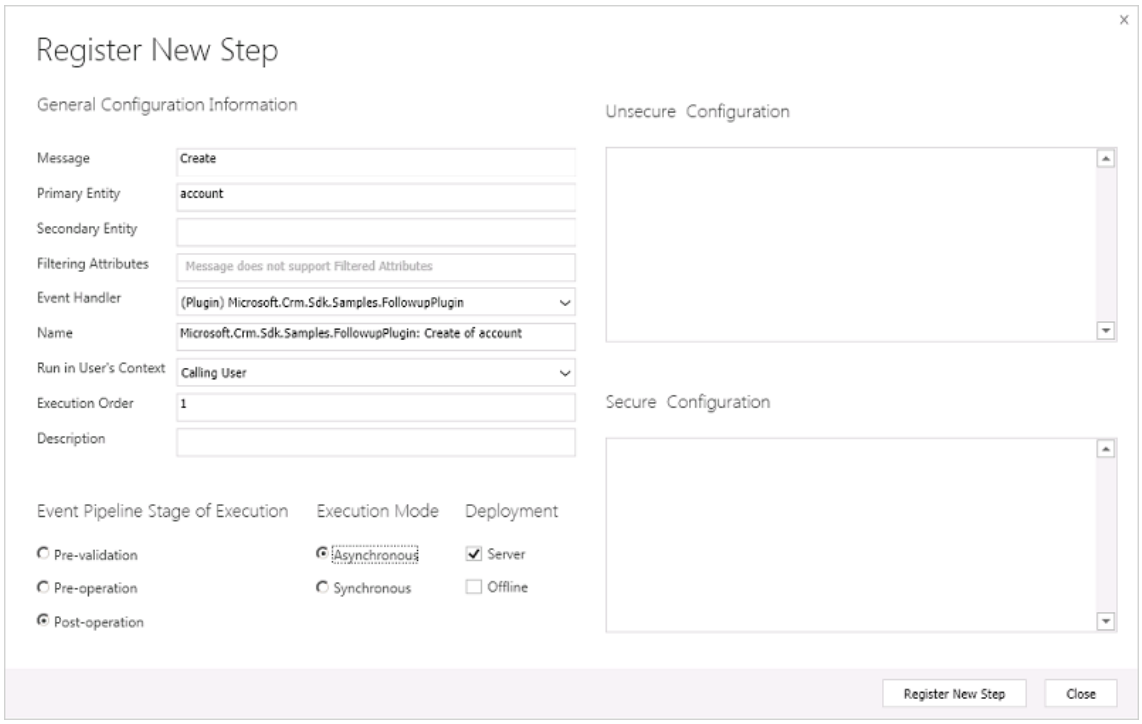
Kuvassa 3 on valittu rekisteröitäväksi assemblyksi SamplePlugin.dll. Kuvassa näkyy myös lista plugineista, jotka ovat assemblyssä. Pluginlistasta näkyy, että vain yksi plugin, ”Microsoft.Crm.Sdk.Samples.FollowupPlugin”, on valittu rekisteröiräväksi. Rekisteröinti käynnistyy ”Register Selected Plugins” -painikkeesta. Rekisteröintinäkymän alaosassa on myös pieni ikkuna lokille. Siihen tulostuu tietoja rekisteröinnin suorituksesta.

Kuvasta 3 voidaan lukea assemblyn muokattavissa olevat tiedot. Nämä ovat Isolation mode ja tallennuspaikka. Isolation mode määrittää, ajetaanko pluginia rajoittamattomilla oikeuksilla vai sandboxissa, eli rajoitetuin oikeuksin. Mahdollisia tallennuspaikkoja ovat tietokanta, paikallinen levy sekä GAC (Global Assembly Cache). Isolation moden ja tallennuspaikan lisäksi assemblystä valitaan, mitkä pluginit halutaan rekisteröidä. Valinta tehdään kuvassa 3 näkyvästä listasta ruksaamalla halutut pluginit.

2.3.1 Sovelluslaajennoksien herätteet

Plugin sisältää ohjelmointilogiikan, mutta se tarvitsee toimiakseen vähintään yhden stepin. Steppi on heräte, josta plugin käynnistyy. Steppejä voi lisätä plugineille Plugin Registration Toolilla ja niitä voi luoda pluginille mielivaltaisen määrän.

Stepin voi ajatella olevan eräänlainen keskeytyspalvelu. Plugin taas on logiikka, joka suoritetaan keskeytyksen tapahtuessa. Esimerkki stepin rekisteröimisestä on kuvassa 4.



KUVA 4. Stepin rekisteröinti (Microsoft 2018)

Stepillä on monta eri asetusta. Asetusten avulle määritetään tarkasti, minkälaisesta tapahtumasta plugin käynnistetään sekä missä vaiheessa se suoritetaan. Taulukossa 1 on listattuna stepin muokattavissa olevat ominaisuudet ja niiden selitykset.

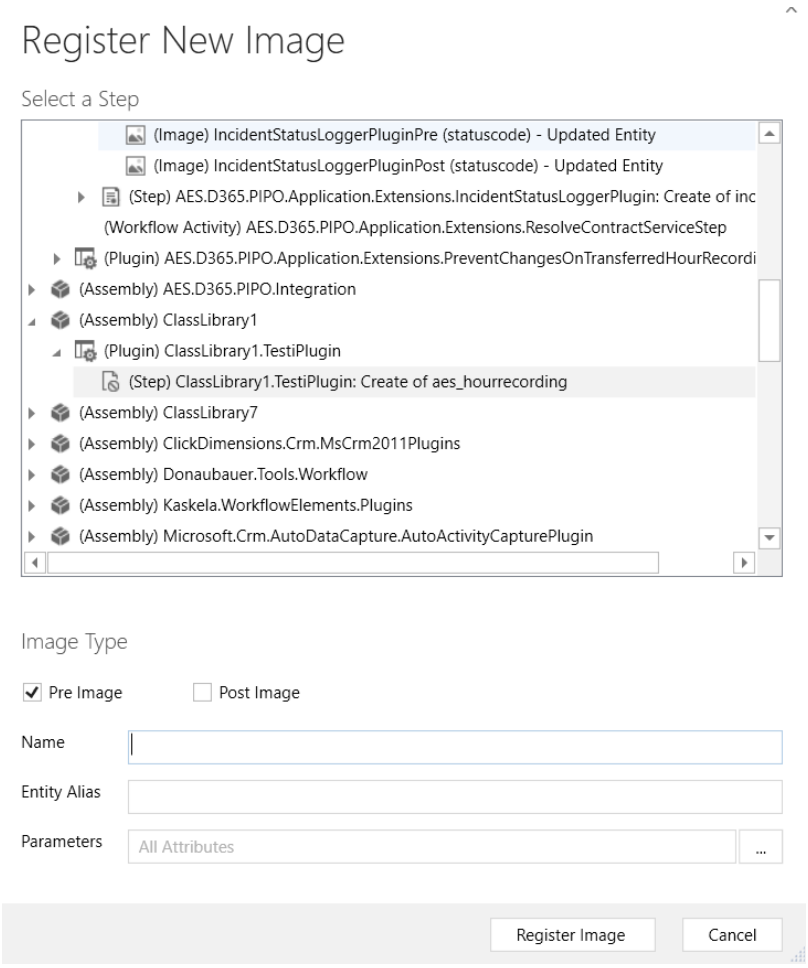
TAULUKKO 1. Pluginstepin ominaisuudet

Ominaisuus	Selite
Message	Määrittää mistä tapahtumasta plugin käynnistetään. Usein käytettyjä ovat esim. “Create”, “Update” ja “Delete”
Primary Entity	Määrittää ensisijaisen entiteetin, johon tapahtuma sidotaan. Voi olla tyhjä, jolloin tapahtuma sidotaan kaikkiin entiteetteihin.
Secondary Entity	Toissijainen entiteetti. Käytetään vain viestien “SetRelated” ja “RemoveRelated” kanssa, jotka ovat vanhentuneita.
Filtering Attributes	Määrittää, mistä entiteetin tiedoista plugin käynnistetään. Hyödyllinen esim. Update-viestin kanssa, jos plugin halutaan suorittaa vain tietyn tiedon päivityksestä.
Event Handler	Tapahtuman käsittelijä, eli plugin, joka suoritetaan.
Name	Stepin nimi. Plugin Registration Tool generoi kuvaavan nimen automaattisesti.
Run in User’s Context	Minkä käyttäjän kontekstissa plugin suoritetaan. Joko tietty käyttäjä tai kutsuva käyttäjä.
Execution Order	Suoritusprioriteetti. 1 on suurin prioriteetti. Suurimman prioriteetin pluginit suoritetaan ennen pienemmän prioriteetin plugineja.
Description	Valinnainen kuvaus. Plugin Registration Tool asettaa oletuksena samaan arvoon kuin nimi.
Event Pipeline Stage of Execution	Missä vaiheessa plugin suoritetaan. Vaihtoehdot ovat Pre-validation, Pre-operation ja Post-operation. Eniten käytetyt ovat Pre- ja Post-operation.
Execution Mode	Synkronisen tai asynkronisen suorituksen valinta
Deployment	Suoritetaanko plugin serverillä, Outlook clientissä vai molemmissa.

2.3.2 Entiteettikuvat

Oletuksena pluginin suorituskontekstissa on vain entiteettin muutetut tiedot. Joskus on tarvetta olla käsiteltävissä myös tiedot ennen tapahtumaa tai tapahtuman jälkeen. Esimerkiksi voidaan haluta tapahtuvan toiminto, kun tietyn kentän arvo muutetaan tietystä arvosta joksikin muuksi tietyksi arvoksi. Tällöin apuna ovat pluginimaget.

Sovelluslaajennoksen entiteettikuvan, eli pluginimagen, sisältönä on entiteetin tiedot joko ennen tai jälkeen pluginin laukaissutta tapahtumaa. Imagen sisältöä on mahdollista muokata asetuksilla. Imageita määritetään plugineille steppikohtaisesti. Stepit eivät tarvitse toimiakseen imagea, vaan niitä rekisteröidään aina käyttötarpeen mukaan. Kuvassa 5 on esimerkki imagen rekisteröinnistä.



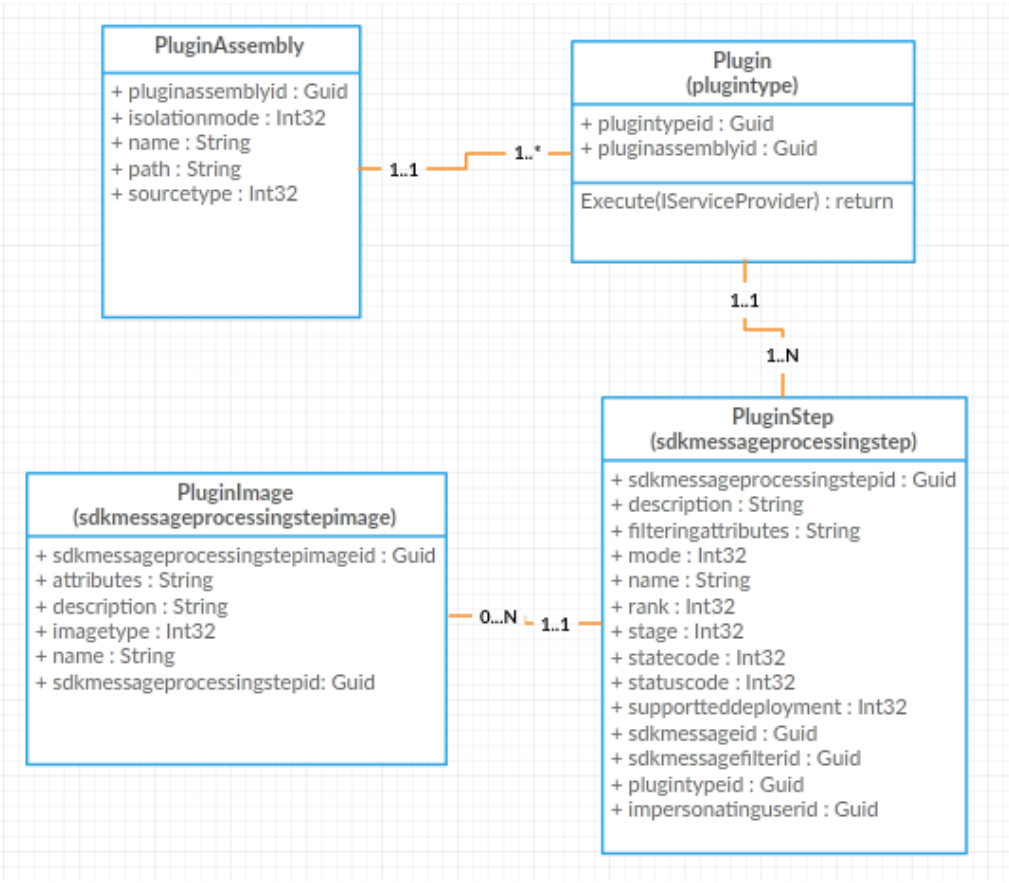
KUVA 5. Stepin rekisteröinti

Kuvassa 5 on listassa valittuna steppi nimeltä ”TestiPlugin: Create of aes_hourrecording”. Imagea siis luodaan TestiPluginin Create-steppiin. Steppi on liitetty

aes_hourrecording-entiteettiin. Kuvan alaosassa on stepin asetukset. Ensimmäinen valinta on Imagen asettamien joko Pre- tai PostImageksi. Imagelle voi myös asettaa nimen ja kuvauksen. Nimellä ja kuvauksella ei ole toiminnallista merkitystä. Järkevä nimeäminen kuitenkin helpottaa asioiden tunnistamista, varsinkin jos organisaatioon on rekisteröity paljon eri plugineja. Imagelle voi myös määritellä, mitä tietoja se sisältää. Jos tiedetään, että pluginissa käytetään vain tiettyjä tietoja, voidaan imagen sisältämää tietomäärää rajoittaa. Tällä tavoin saadaan rajoitettua lähetetyn datan määrää.

2.4 Tietomalli

Pluginkirjastoilla on Microsoftin ennalta määritelty tietorakenne, joka on esitetty kuvassa 6. Rakenne on selvitetty Microsoftin dokumentaatioista. Kuvaan on otettu mukaan vain ne tiedot, jotka ovat muokattavissa. Muokattavien tietojen lisäksi on myös paljon metatietoja, joita asetetaan pluginien rekisteröinnin yhteydessä. Metatietoja ovat esimerkiksi luojan id, muokkaajan id, luontiaika, muokkausaika jne.



KUVA 6. Pluginien pelkistetty rakenne

Kaikissa muissa suhteissa on alarajana 1 paitsi stepin suhteessa imageen on 0. Tällä on haluttu korostaa sitä, että toimivassa konfiguraatiossa on assemblyssä vähintään yksi plugin, jolla on vähintään yksi steppi. Myöskään plugin ei voi olla olemassa ilman assemblyä, eikä steppi ilman pluginia. Assembly ilman pluginia eikä plugin ilman steppejä tee mitään, vaikka sellainen tilanne on teknisesti mahdollinen. Tälläisen tilanteen katsotaan kuitenkin olevan konfiguraatiovirhe.

Kuvassa 6 luokkien otsikkokentässä on ensin luokan kutsumanimi ja sen alla suluissa luokkien tietokantanimet. Kutsumanimet ovat niitä, joilla luokkia kutsutaan dokumentaatioissa tai malliluokissa. Tietokantanimet ovat niitä, joita käytetään CRM:n tietokannassa. Tietokantanimet ovat yleensä piilotettu käyttäjältä, mutta on hyvä tietää, että luokilla on eri kutsumanimet ja tietokantanimet.

Luokkien muuttujat ovat tyypiltään joko guid, int32 tai string. Guid-tyyppisset muuttujat ovat id-kenttiä, jotka yksilöivät objektit. Luokkien pääavaimet ovat nimetty syntaksilla <luokan nimi>id. Luokilla on myös vierasavaimia, joilla toteutetaan luokkien väliset relaatiot. Int32-tyyppiset muuttujat ovat lähes kaikki erilaisia valintoja. Valintojen eri arvoja kuvataan kokonaisluvuilla. Poikkeuksena on stepin Rank-muuttuja, johon syötetään kokonaisluku, joka kuvaa stepin prioriteettia. String-tyyppiset muuttujat ovat vapaita tekstikenttiä. Käyttäjä voi syöttää niiden arvoksi mielivaltaisen merkkijonon. String-muuttujissa säilötään objektia kuvaavaa dataa, joka ei ole merkityksellistä pluginien suorittamisen kannalta.

3 SUUNNITTELU

3.1 Ohjelmointiympäristö

Ohjelmointiympäristö käytetään Visual Studio 2017:ää. Ohjelmointikielenä on C#. Sitä käytetään siksi, että yrityksessä pluginien kehityksessä käytetään yleisesti C#:ia. Yksi ohjelman tavoitteista on konfiguraatiorajapinnan tekeminen, joten on järkevää tehdä ohjelma ja rajapinnat samalla kielellä kuin millä pluginitkin tehdään. Kieli on kaikille kehittäjille tuttu ja se on jatkokehityksen kannalta hyvä asia.

Visual Studiossa on käytössä myös Team Foundation Server (TFS). Se tarjoaa kehitystyön tueksi erilaisia työkaluja, mutta ennen kaikkea TFS on versionhallintaohjelma. Sitä käyttämällä koodeista tallennetaan historiatietoja ja uusimmat koodit ovat käyttäjien saatavilla helposti ja nopeasti. (Microsoft 2019)

3.2 IPlugin-rajapinta

IPlugin-rajapinta on Microsoftin tarjoama rajapinta pluginien kehitykseen. Se löytyy kirjastosta nimeltä Microsoft.Xrm.Sdk. (Microsoft N.d.) IPlugin-rajapinta on hyvin yksinkertainen. Se sisältää ainoastaan metodin Execute(IServiceProvider). Execute-metodi on pluginin ns. entry point. Se on ensimmäinen asia, joka pluginista suoritetaan. Normaalisti entry point on C#-ohjelmissa Main-niminen metodi.

Execute-metodin parametri IServiceProvicer on säiliö palveluolioille. Se sisältää suorituskontekstin, organisaatiopalvelun ja muuta hyödyllistä tietoa tapahtumasta, joka käynnisti pluginin. (Microsoft 2016).

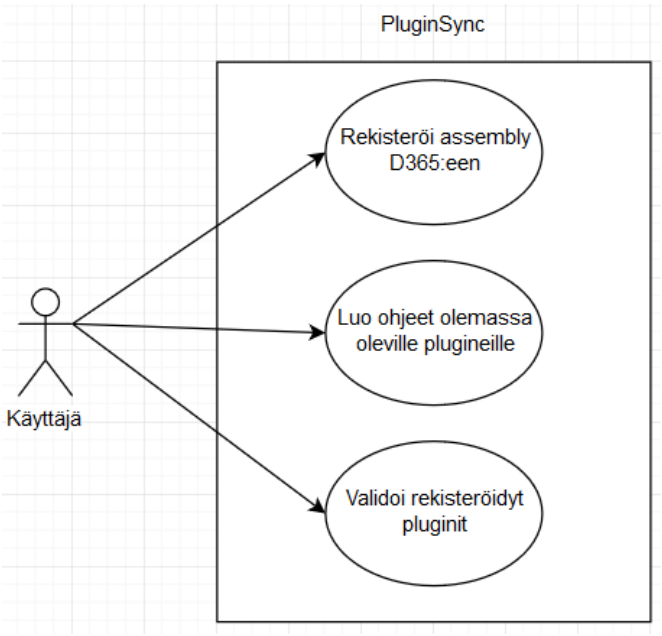
3.3 Määritelmät

Accountor Enterprise Solutions Oy on määritellyt, mitä toteutettavan ohjelman tulee tehdä.

> Ydintarve on saada plugin rekisteröityä ilman manuaalisia välivaiheita automaattisella työkalulla oikeaan CRM-organisaatioon. Tavoitteena on nopeuttaa kehitysprosessia, pienentää PluginRegistrationToolilla käsityönä

tehtävien päivitysten aiheuttamaa inhimillisten virheiden mahdollisuutta sekä parantaa pluginien koodin dokumentoitavuutta. Ajo pitää pystyä skriptaamaan komentoriviltä ajettavaksi, jotta ajo pystytään automatisoimaan.

PluginSyncille määriteltiin kolme eri käyttötapausta. Käyttötapauksista tehtiin UML-kaavio, joka on kuvassa 7.



KUVA 7. PluginSyncin käyttötapaukset

Ensimmäinen, ja tärkein, käyttötapaus on pluginassemblyjen rekisteröinti D365:een. Ideana on, että käyttäjä voi ajaa ohjelman komentorivisovelluksena. Parametrina annetaan rekisteröitävä assembly. PluginSync osaa annetun assemblyn perustella rekisteröidä sen sisältämät pluginit CRM:ään.

Toinen käyttötapaus on rekisteröintiohjeiden luonti olemassa oleville plugineille. Ennen PluginSyncin toteutusta on ollut paljon suljettuja tai edelleen jatkuvia projekteja. Näissä vanhoissa projekteissa on voitu käyttää plugineja. Pluginien määrä per projekti vaihtelee muutamista useisiin kymmeniin. Tästä syystä tarvitaan toiminto, jolla voidaan luoda rekisteröintiohjeet vanhoille plugineille. Käsin ohjeiden luonti vanhoille plugineille olisi liian työlästä ja virhealtista.

Kolmas käyttötapaus on rekisteröityjen plugineiden validointi. Tällä toiminnolla voi nopeasti tarkistaa, vastaako rekisteröityjen plugineiden konfiguraatio sitä, mitä on

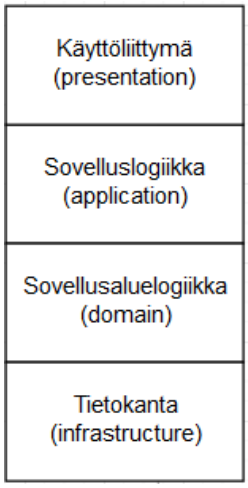
määritelty rekisteröintiohjeissa. Tämän avulla voidaan esimerkiksi virhetilanteissa tarkistaa, onko pluginien konfiguraatio oikein. Konfiguraatiovirheiden tunnistaminen virheiden selvityksessä voi säästää huomattavasti aikaa.

3.4 Arkkitehtuuri

Työn arkkitehtuurina käytetään kerrosarkkitehtuuria. Sen perusajatuksena on, tietyllä tasolla oleva komponentti tai yksittäinen palvelu toteutetaan käyttäen alemman kerroksen tarjoamia komponentteja tai palveluita (Koskimies & Mikkonen 2005, 126).

Kerrosarkkitehtuuri koostuu tasoista, jotka on järjestetty jonkin abstrahointiperiaatteen mukaan nousevaan järjestykseen. Yleinen skaala on laite/ihminen. Tällöin alemmat tasot tarjoavat laitetta tai käyttöjärjestelmää lähellä olevia toimintoja ja ylemmät tasot taas käyttöliittymän ja sovelluslogiikan. (Koskimies & Mikkonen 2005, 126).

Tässä työssä toteutetaan nelikerroksinen arkkitehtuuri. Kerrokset ovat alhaalta ylöspäin infrastruktuuri-, sovellusalue-, sovellus- ja käyttöliittymäkerros. Kerrokset ovat kuvattuna kuvassa 8.



KUVA 8. Arkkitehtuurin kerrokset (Koskimies & Mikkonen 2005, 128)

Tietokanta- eli infrastruktuurikerros tarjoaa yleistä infrastruktuuritukea. Kerroksen vastuulla on assemblyjen reflektioon liittyviä asioita sekä CRM-organisaation kanssa kommunikointia.

Sovellusaluelogiikka- eli liiketoimintakerros tarjoaa sovelluksen kaikille alueille yhteisiä toimintoja. Liiketoimintalogiikkaa on tämän työkaluin osalta esimerkiksi pluginien rekisteröintiohjeiden selvittäminen assemblystä ja näiden rekisteröintiohjeiden perusteella rekisteröinti CRM-organisaatioon.

Sovelluslogiikkakerros tarjoaa yksittäisen sovelluksen logiikan. Sen vastuulla on suorituksen ohjaaminen. Tätä kerrosta vastaan rakennetaan komentorivisovellus ja mahdolliset tulevat käyttöliittymät.

Käyttöliittymäkerros tarjoaa yksittäisen sovelluksen käyttöliittymän. Tällä kerrokselle ei tule varsinaista sovelluslogiikkaa. Kerros vain näyttää halutun käyttöliittymän ja osaa käyttäjän syötteiden perusteella kutsua varsinaista sovelluslogiikkaa alemmilta kerroksilta. Tulevaisuudessa työkalulle saatetaan toteuttaa muitakin sovelluksia, joten tässä kerrokselle on mahdollisimman vähän sovelluslogiikkaa.

4 TOTEUTUS

PluginSync toteutuksen pohjana käytetään Microsoftin julkaisemaa lähdekoodia Plugin Registration Tool 2013:sta.

PluginSync toteutetaan konsoliohjelmana. Ohjelman suoritusta ohjaillaan sille syötettävillä parametreillä. Parametrisointi mahdollistaa ohjelman suorittamisen skriptien avulla. Skripteillä voidaan automatisoida ohjelman suoritus ja se mahdollistaa esimerkiksi ohjelman ajamisen aina pluginkoodin kääntämisen jälkeen.

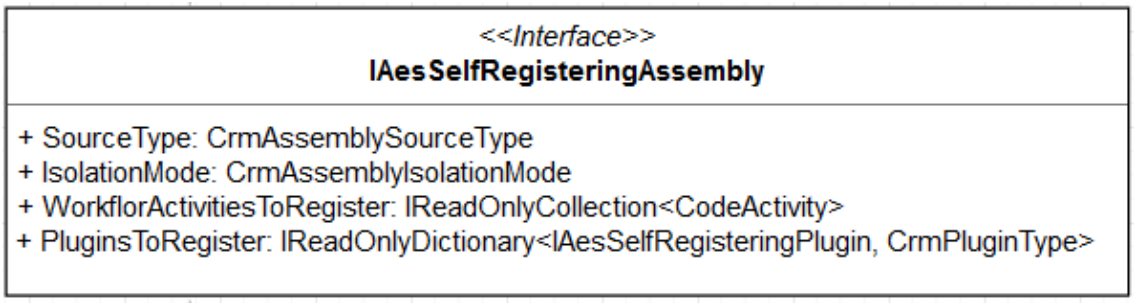
4.1 Rajapinnat

Konsoliohjelman lisäksi toteutetaan rajapinnat pluginien eri osille. Tarvittavat osat, jotka ovat esiteltynä tarkemmin luvussa 2.4, ovat Assembly, Plugin, Step ja Image. Rajapinnoilla määritetään kunkin osan tarvitsemat tiedot.

Rajapintojen ideana on toteuttaa yksikäsitteinen tietorakenne, jota PluginSync käyttää pluginien konfiguroimiseen. Rajapintojen avulla käyttäjä voi nopeasti tehdä konfiguraation pluginille.

4.1.1 Assembly-rajapinta

Assemblyille toteutetaan rajapinta nimeltä IAesSelfRegisteringAssembly. Rajapinta määrittelee tarvittavat asetukset, joilla assembly saadaan rekisteröityä D365:een. Toteutettava rajapinta on kuvassa 9.



KUVA 9. IAesSelfRegisteringAssembly-rajapinta

IAesSelfregisteringAssembly-rajapinnalla on neljä ominaisuutta. Assemblyllä on muitakin ominaisuuksia kuin mitä kuvassa näkyy. Toteutetun rajapinnan ominaisuudet ovat sellaisia, joita pystyy muokkaamaan.

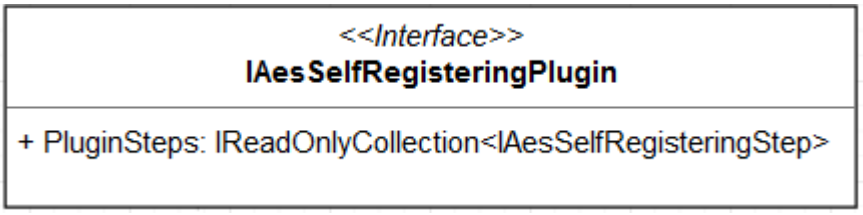
SourceType on assemblyn tallennuspaikka. Mahdollisia tallennuspaikkoja ovat tietokanta, paikallinen levy ja GAC. SourceTypen tyyppi on CrmAssemblySourceType. Se on lueteltu tyyppi, jonka mahdolliset arvot kuvaavat eri tallennuspaikkoja.

IsolationMode on valinta, että suoritetaanko assemblyn pluginit rajoitetuin vai täysin oikeuksin. Sen tyyppi on CrmAssemblyIsolationMode, joka on lueteltu tyyppi. Sen mahdolliset arvot kuvaavat kahta eri valittavaa arvoa.

PluginsToRegister on lista rekisteröitävistä plugineista ja WorkflowActivitiesToRegister on lista rekisteröitävistä Workflowactivityistä. Oikeasti Workflow activity ja plugin eroavat toisistaan, mutta tässä yhteydessä workflow activityn voi ajatella olevan eräänlainen plugin. Tässä työssä toteutettava ohjelma toteuttaa vain pluginien rekisteröimisen. WorkflowActivitiesToRegister on luotu tulevaisuutta varten.

4.1.2 Plugin-rajapinta

Plugineille toteutettiin rajapinta nimeltä IAesSelfRegisteringPlugin. Rajapinta määrittää, mitä asetuksia pluginille voidaan asettaa. Toteutettu rajapinta on kuvassa 10.

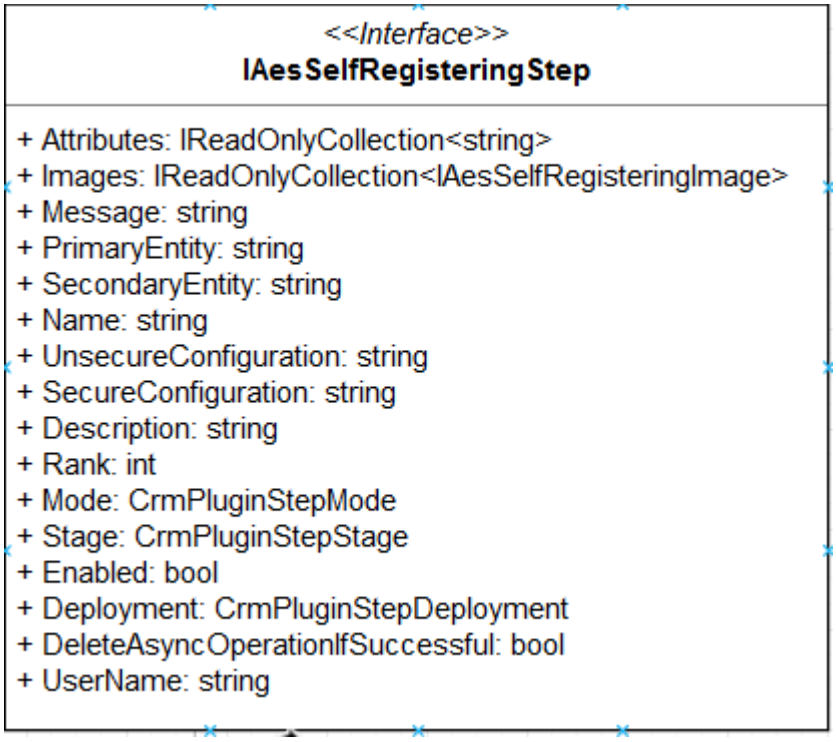


KUVA 10. IAesSelfRegisteringPlugin-rajapinta

Plugineilla ei ole muokattavia asetuksia. Pluginit sisältävät vain halutun logiikan. Pluginin käynnistämistä koskevat säännöt ovat määritelty pluginin stepeissä. Pluginin täytyy siis tietää, mitä steppejä sille rekisteröidään. Tämän vuoksi rajapintaan on tehty ainoastaan lista, jonka sisältönä on pluginin stepit.

4.1.3 Step-rajapinta

Stepeille toteutettiin rajapinta nimeltä IAesSelfRegisteringStep. Tämä rajapinta on kuvassa 11.

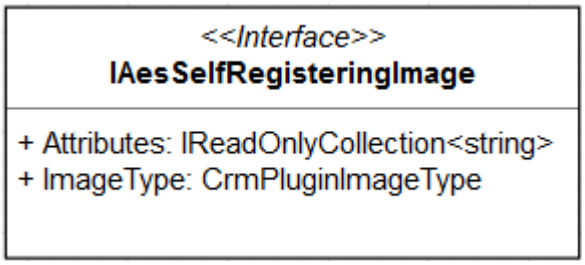


KUVA 11. IAesSelfRegisteringStep-rajapinta

Stepeillä on paljon muokattavia asetuksia, jotka toteutettiin myös rajapintaan. Pluginstepin asetukset pääasiassa määrittävät, mistä tapahtumasta plugin käynnistetään. Asetuksista on kerrottu tarkemmin taulukossa 1 luvussa 2.3.1.

4.1.4 Image-rajapinta

Imageille tehtiin rajapinta nimeltä IAesSelfRegisteringImage. Toteutettu rajapinta on kuvassa 12.



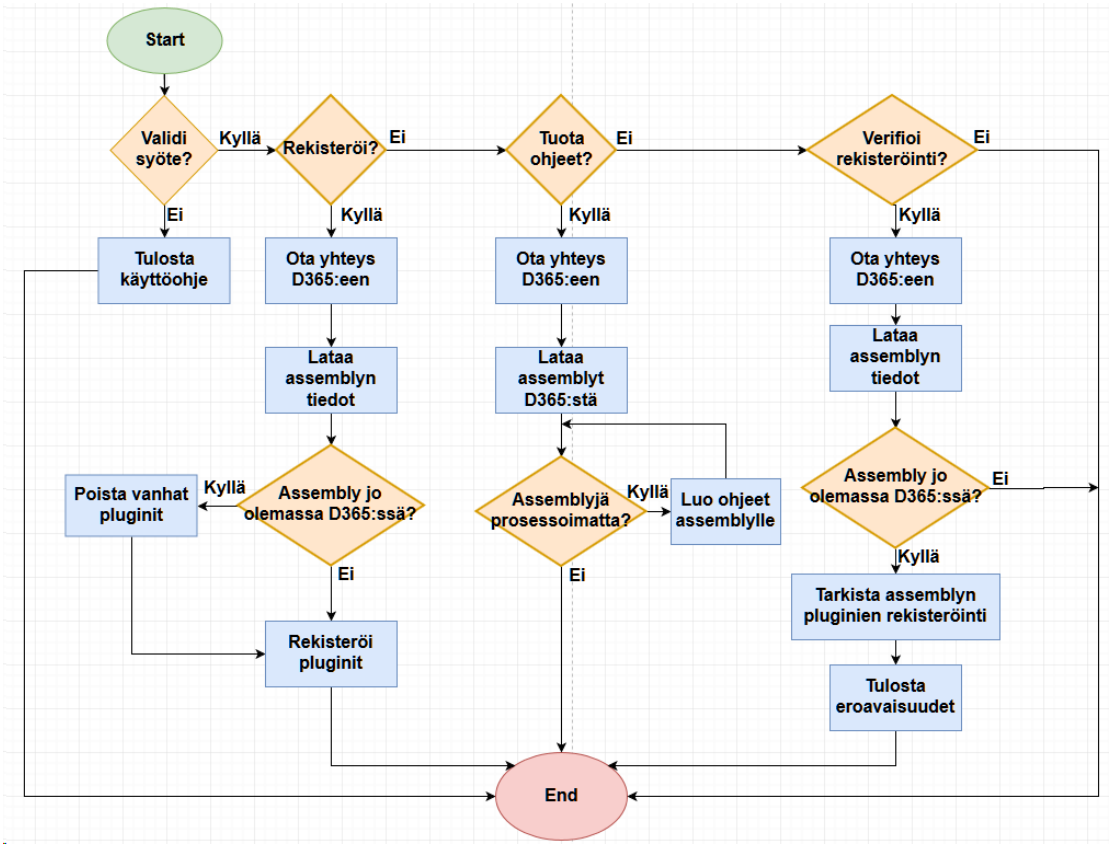
KUVA 12. IAesSelfRegisteringStep-rajapinta

Rajapintaan toteutettavat asetukset ovat ”Attributes”, joka määrittää, mitkä tiedot imageen laitetaan, sekä ”ImageType”, joka määrittää, otetaanko tiedot ennen vai jälkeen stepin määrittelevän tapahtuman tapahtumista.

Imageille on kuvassa 12 olevien asetusten lisäksi mahdollista määrittää mielivaltainen nimi sekä kuvaus. Imaget haetaan pluginien koodissa niiden nimen perusteella, joten imageiden nimien halutaan olevan tietynlaisia. Tämän takia rajapintaan ei toteuteta nimen ja kuvauksen asettamista. Sen sijaan imagelle luodaan nimi PluginSyncin koodissa. Imagen nimi ja kuvaus luodaan syntaksilla <PlugininNimi><Pre/Post>. Esimerkiksi pluginille ”TestiPlugin” eräs imagen nimi voisi olla ”TestiPluginPre”.

4.2 Suorituslogiikka

Luvussa 4.2 ohjelmalle määriteltiin kolme käyttötapausta: rekisteröinti, validointi ja ohjeiden luonti. Ohjelma toteutettiin niin, että käyttäjä voi syöttää parametrina, minkä toiminnon haluaa ohjelman suorittavan. Ohjelman suorituslogiikka on kuvassa 13.



KUVA 13. Ohjelman karkea logiikka

Ohjelman suoritus alkaa käyttäjän syötteen tarkastamisella. Jos se on epävalidi, tulostetaan käyttäjälle ohjelman käyttöohjeet. Jos syöte on validi, tarkistetaan minkä toiminnon käyttäjä haluaa suorittaa, jonka jälkeen haluttu toiminta suoritetaan.

Ohjelman logiikka on jaettu kolmeen haaraan. Kukin haara toteuttaa yhden ohjelman käyttötapauksen.

Toteutettu ohjelma on konsoliohjelma, joten sillä ei ole graafista käyttöliittymää. Käyttäjä ei muutenkaan voi vaikuttaa ohjelmaan suoritukseen kesken suorituksen. Käyttäjä valitsee suoritettavan toiminnon syöttämällä ohjelmalle oikeat parametrit.

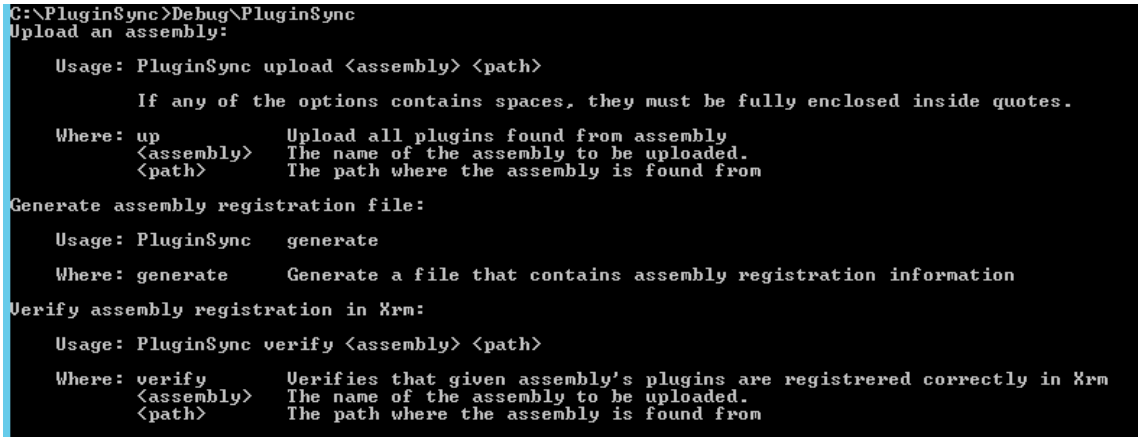
Ohjelma suoritetaan joko käsin komentorivin kautta tai suorittamalla skriptin, jossa on ajokomennot. Ohjelman käynnistävän komennon syntaksi on ”PluginSync <toiminto> <assemblynNimi> <assemblynPolku>”.

Komennon kohdalla ”toiminto” on kolme eri vaihtoehtoa. Jokainen niistä totetuttaa yhden käyttötapauksen. Mahdolliset arvot ovat ”upload”, ”generate” ja ”verify”. Upload-toiminolla rekisteröidään haluttu assembly. Generate-toiminto tuottaa halutun

organisaation assemblyita rekisteröintitiedostot. Verify-toiminlla tarkistetaan, vastaako halutun assemblyn ohjeet sitä, miten jonkin organisaation pluginit ovat konfiguroitu.

Esimerkiksi jos haluttaisiin rekisteröidä C:n juuressa oleva assembly nimeltä TestAssembly, olisi komento ”PluginSync upload TestAssembly C:\”. Jos taas haluttaisiin luoda organisaation plugineista ohjeet, olisi komento ”PluginSync generate”. Edellä olevien komentojen tapauksessa oletetaan, että ne syötetään komentoriviltä ja komentorivillä on navigoitu kansioon, jossa PluginSyncin ajotiedosto sijaitsee.

Jos Pluginsyncille ei syötetä mitään parametreja tai parametrit on syötetty väärin, tulostetaan ruudulle ohjelman käyttöohjeet. Käyttöohjeiden tulostaminen on käyttäjälle hyödyllisempää kuin pelkän ilmoituksen näyttäminen, jossa sanotaan, että on tapahtunut virhe. Tulostettu ohje on kuvassa 14.



KUVA 14. PluginSyncin käyttöohjeet

Haluttu organisaatio valitaan ohjelman app.config-tiedostossa. App.config on C#-ohjelmien oletuskonfiguraatiotiedosto. CRM:n yhteystiedot asetetaan tuohon tiedostoon. Tiedoston tietotyyppinä xml. Yhteystiedot listataan connectionStrings-elementin sisään. Esimerkiksi yhteystieto voisi olla

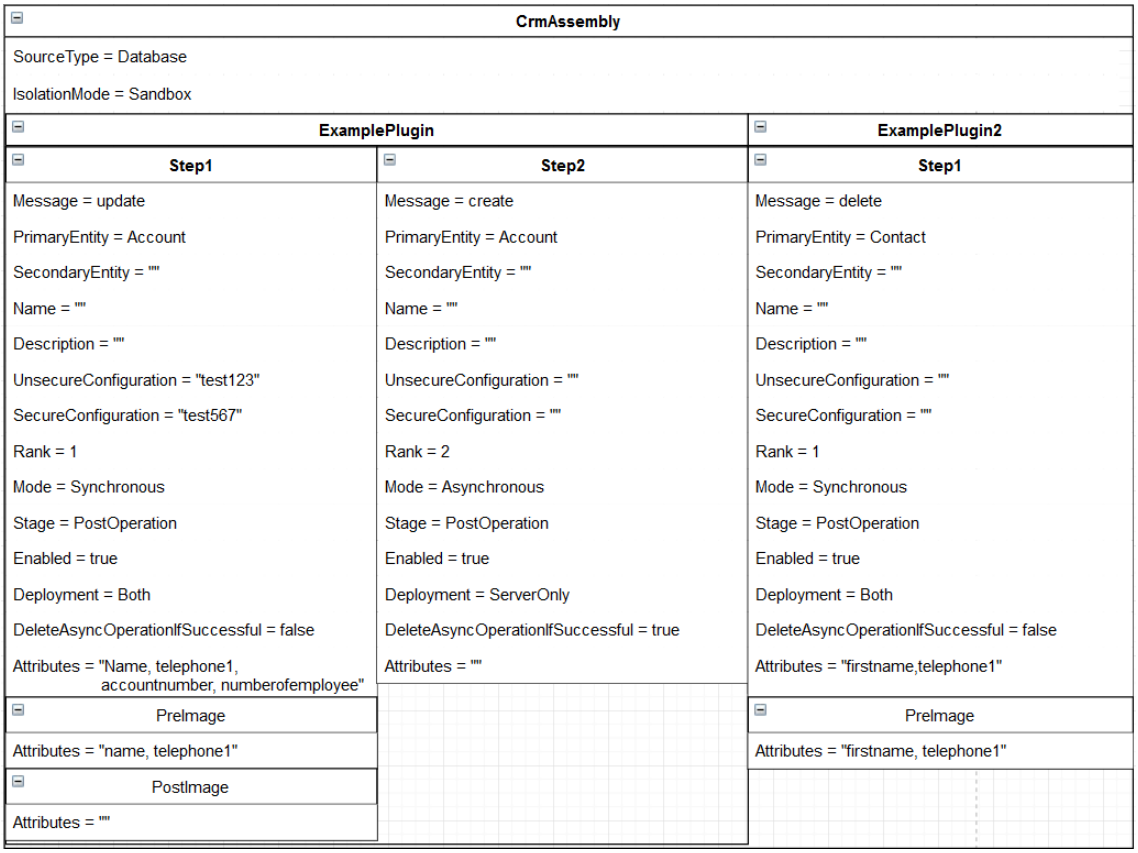
```
<connectionStrings>
        <add name=”CRM” connectionString=”AuthType=AD;URL=.” />
</connectionStrings>
```

5 TESTAUS

Ohjelmaa testattiin virtuaalikoneella. Virtuaalikoneen käyttöjärjestelmä oli Windows Server 2012 ja sen CRM versiona oli CRM 2016. Toteutetun ohjelman oli tarkoitus toimia pääasiassa D365:llä, mutta ohjelman logiikan voi hyvin testata vanhemmallakin CRM versiolla. Haittana on se, että CRM 2016:lla testatessa ei voida todeta, että saako ohjelma yhteyden muodostettua D365:een. Logiikan testaaminen oli huomattavasti tärkeämpää kuin yhteyksien testaaminen. Mahdolliset yhteysongelmat voidaan ratkaista tulevaisuudessa, jos ohjelmaa päätetään jatkokehittää.

Ohjelman ajotiedosto ja sen tarvitsemat kirjastot vietiin virtuaalikoneelle polkuun C:\PluginSync\Debug. Rekisteröitävä testiassembly vietiin polkuun C:\PluginSync. Ohjelmaa testattiin käynnistämällä se komentorivin kautta. Komentorivillä oli navigoitu polkuun C:\PluginSync.

Testausta varten luotiin testiassembly, jonka sisältönä oli kaksi pluginia. Pluginilla 1 oli kaksi steppiä ja pluginilla 2 oli yksi steppi. Testiassemblyn konfiguraatio tehtiin sellaiseksi, että siinä ei ollut virheitä. Testauksella haluttiin selvittää, että pystyykö ohjelma rekisteröimään plugineja CRM:ään. Ohjelma ei tarkista konfiguraation oikeellisuutta, vaan oikeanlaisen konfiguraation luominen jää käyttäjän vastuulle. Assemblyn sisältö on listattu tarkemmin kuvassa 15.



KUVA 15. Testiassemblyn sisältö

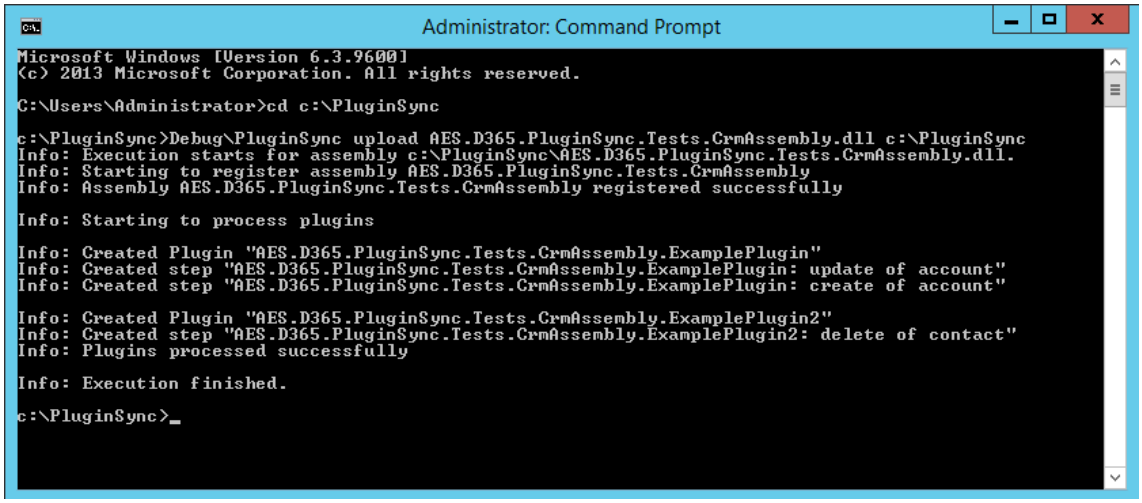
Kuvassa 15 näkyy testiassemblyn haluttu konfiguraatio. Assemblytason asetukset ovat kuvassa ylimpänä. Ne ovat SourceType, jonka arvoksi haluttiin Database, ja IsolationMode, jonka arvoksi haluttiin Sandbox.

Assemblyssä on kaksi pluginia nimiltään ”ExamplePlugin” ja ”ExamplePlugin2”. Plugineilla ei ole itsessään konfiguroitavia asetuksia. Plugineille on tehty steppejä, jotka määrittävät, minkälaisista tapahtumista pluginit käynnistetään. Ensimmäisellä pluginilla on kaksi steppiä ja toisella plugnilla on yksi steppi. Ensimmäisen pluginin ensimmäiselle stepilla ja toisen plugin stepille on tehty imageja. Preimaget sisältävät tietueen tietoja ennen tapahtuman tapahtumista ja postimage tapahtuman tapahtumisen jälkeen.

5.1 Rekisteröinti

Ensimmäiseksi testattiin ohjelman tärkein toiminto, eli pluginien rekisteröinti. Testissä ohjelmalle syötettiin kuvan 14 testiassembly. Rekisteröinnin jälkeen tarkistettiin rekisteröityjen pluginien konfiguraatio Plugin Registration Toolin avulla. Hyväksyttynä

lopputuloksena CRM:ään on rekisteröitynyt kaksi pluginia kuvan 14 mukaisin konfiguraatioin. Rekisteröinnin testiajo on kuvassa 16.

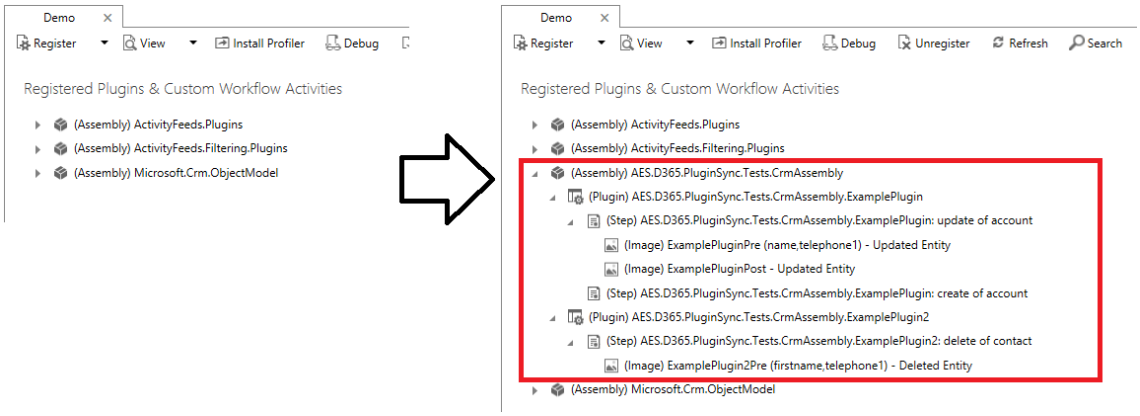


KUVA 16. Rekisteröinnin testiajo

Kuvassa 16 näkyy loki ohjelman suorituksesta. Ensimmäisellä komennolla ”cd c:\PluginSync” siirrytään oikeaan kansioon. Ohjelman käynnistävä komento on ”Debug\PluginSync upload AES.D365.PluginSync.Tests.CrmAssembly.dll c:\PluginSync”. Komento käynnistää pluginsyncin ja kertoo sille, että sen halutaan rekisteröivän haluttu assembly kansiosta C:\PluginSync. Käskyn antamisen jälkeen näytölle tulostuu tietoa ohjelman etenemisestä.

Ohjelman on ensimmäikseksi rekisteröinyt annetun assemblyn. Tämän jälkeen se on rekisteröinyt pluginit yksi kerrallaan. Lopuksi ohjelma on tulostanut näytölle ”Execution finished”, jotta käyttäjä tietää ohjelman suoriutuneen onnistuneesti.

Tarkistetaan Plugin Registration Toolilla rekisteröinnin tuloksia. Kuvassa 17 on tilanne ennen ja jälkeen ohjelman ajamisen.

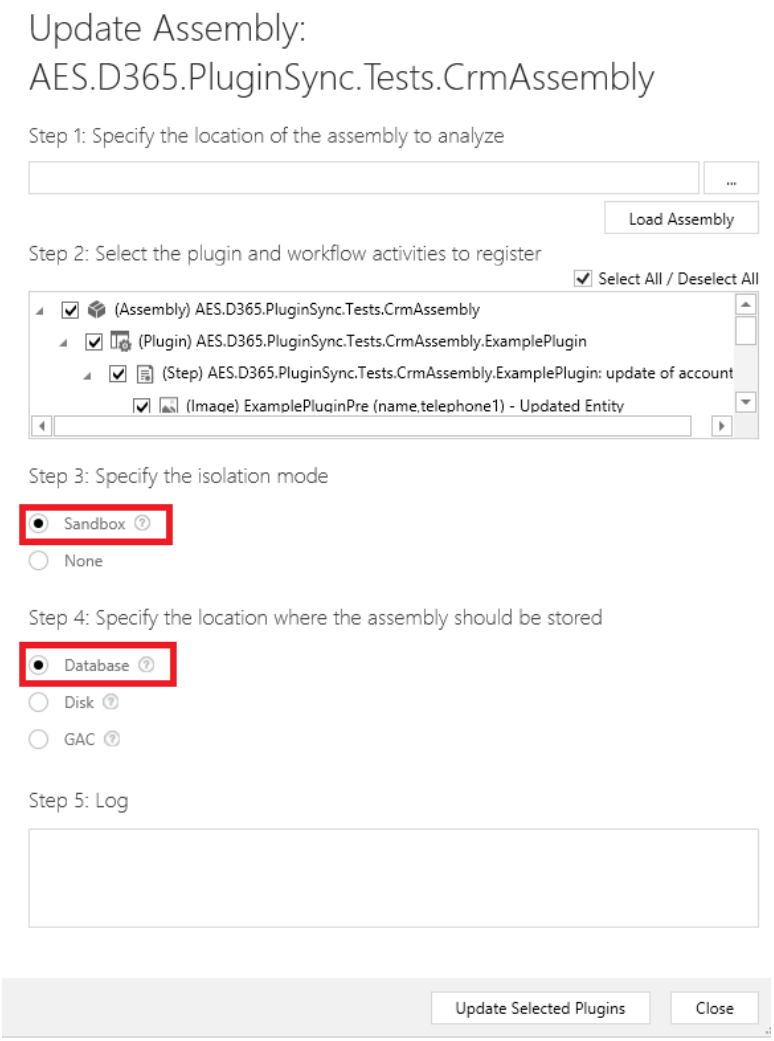


KUVA 17. Demo-organisaation pluginit ennen ja jälkeen ohjelman suorituksen

Kuvan 17 vasemmassa reunassa on tilanne ennen ohjelman ajamista. Organisaatiossa on rekisteröitynä kolme assemblya: ActivityFeeds.Plugins, ActivityFeeds.Filtering.Plugins ja Microsoft.Crm.ObjectModel. Ne ovat oletusassemblyjä, jotka ovat kaikissa organisaatioissa.

Kuvan oikealla puolella on tilanne ohjelman suorittamisen jälkeen. Organisaatiossa on edelleen edellisessä kappaleessa mainitut kolme assemblyä. Niiden lisäksi organisaatiossa on myös AES.D365.PluginSyncs.Tests.CrmAssembly, joka on ohjelmalle syötetty assembly. Sillä on kaksi pluginia. Ensimmäisellä pluginilla on kaksi steppiä ja toisella pluginilla yksi steppi. Ohjelma on siis rekisteröinyt oikean määrän asioita.

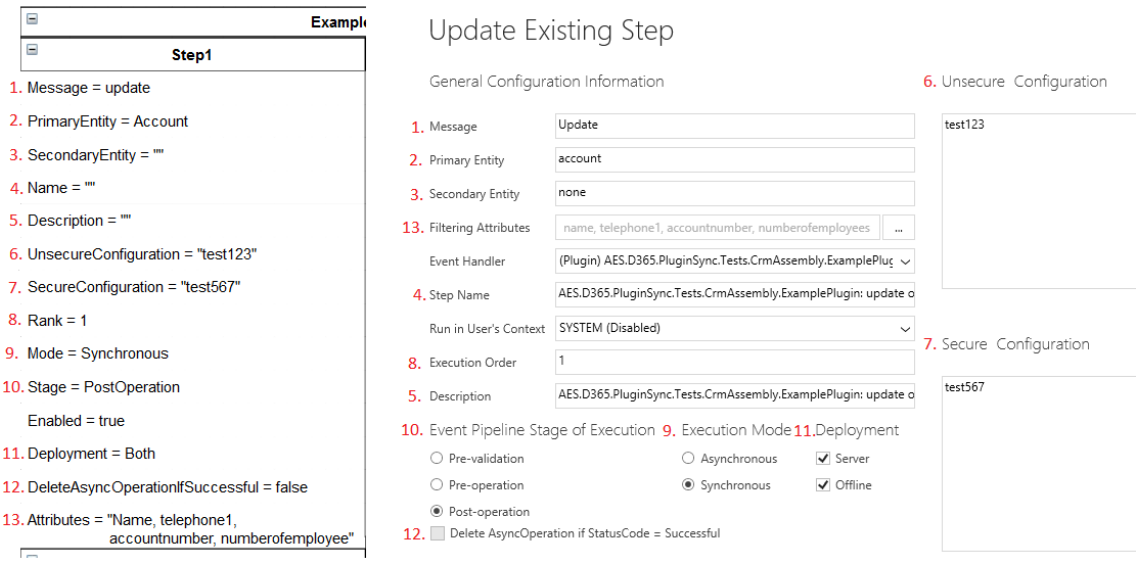
Päälliseltä puolelta näytti, että pluginit ovat rekisteröityneet ohjeiden mukaisesti. Rekisteröinti kuitenkin tarkistettiin vielä tarkemmin. Kuvassa 18 on assemblyn konfiguraatio.



KUVA 18. Testiassemblyn konfiguraatio

Kuvan 18 asetuksia verrataan kuvan 15 tietoihin. Kuvassa 15 assemblyn SourceTypeksi oli asetettu Database ja IsolationModeksi Sandbox. Kuvaan 18 merkittiin edellä mainittujen asetusten arvot. Ohjelma on rekisteröinyt assemblyn oikein.

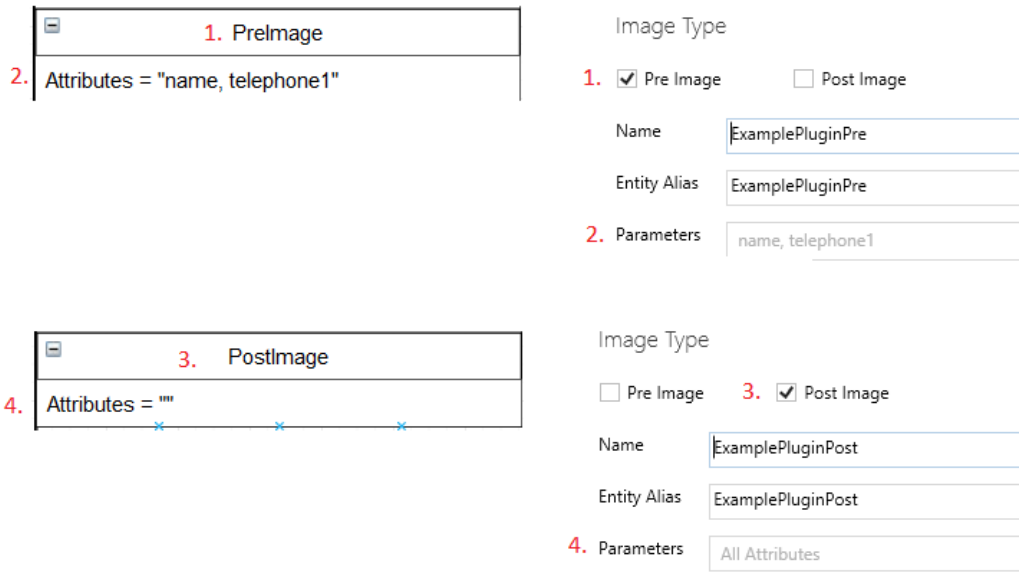
Seuraavaksi tarkistettiin ensimmäisen pluginin ensimmäisen stepin konfiguraatio. Se on kuvassa 19.



KUVA 19. ExamplePluginin Step1

Kuvassa 19 on vasemmalla puolella haluttu konfiguraatio ja oikealla puolella toteutunut konfiguraatio. Toisiaan vastaavat attribuutit on numeroitu. Attribuutit vastaavat toisiaan muuten, mutta nimi ja kuvaus oli halutussa konfiguraatiossa jätetty tyhjiksi. Tässä tilanteessa PluginSync luo nimen ja kuvauksen itse. Nimeämissyntaksi on <Assembly>: <Message> of <Primary Entity>. Kuvan 19 stepin nimeksi tuli AES.D365.PluginSync.Tests.CrmAssembly: update of account, joka on syntaksin mukainen.

Ensimmäisellä stepillä oli myös kaksi imagea. Toinen image oli Preimage ja toinen Postimage. Näiden imageiden konfiguraatio tarkistettiin. Konfiguraatiot ovat kuvassa 20.



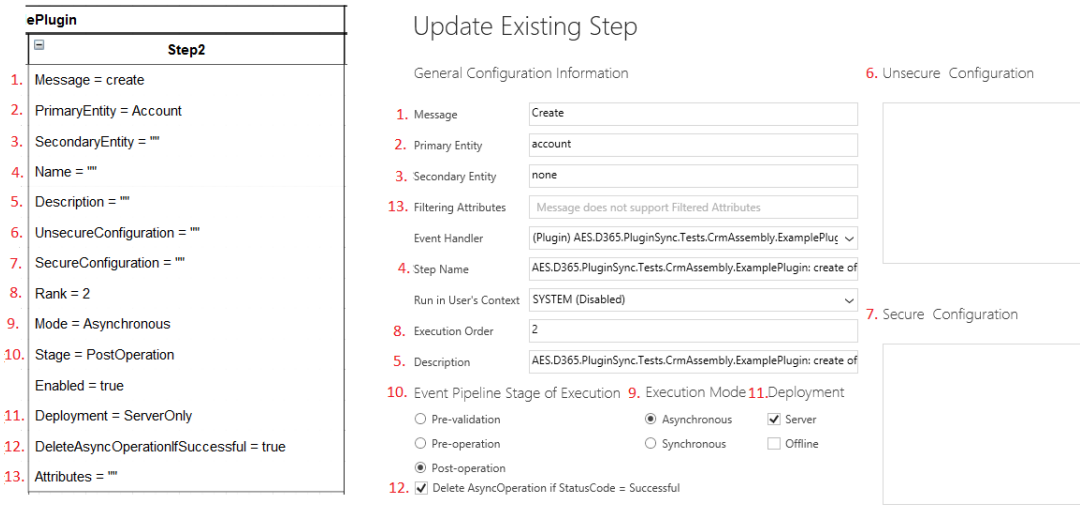
KUVA 20. Stepin 1 imageiden konfiguraatio

Kuvan 20 vasemmalla puolella on halutut konfiguraatiot ja oikealla toteutuneet konfiguraatiot. Yläosassa on image 1 osuus ja alaosassa imagen 2 osuus. Haluttua asetusta vastaava toteutunut asetus on numeroitu vastaavin numeroin.

Kuvassa 20 numerot 1 ja 2 ovat ensimmäisen imagen asetuksia. Image haluttiin preimageksi ja sen attribuuteiksi name ja telephone1. Kuvan oikealta puolelta todetaan, että imagen tyypiksi on valittu Pre Image ja sen Parameterseissa on name ja telephone1. Asetukset ovat siis rekisteröityneet oikein. Name ja Entity Alias ovat ”ExamplePluginPre”. Tämä nimeäminen on luvun 5.1.4 mukainen, eli <PluginiNimi><Pre/Post>. Ensimmäinen image on rekisteröity oikein.

Kuvassa 20 numerot 3 ja 4 ovat toisen imagen asetuksia. Image haluttiin postimageksi ja sen attribuutit jätettiin tyhjäksi, joka vastaa valintaa kaikki attribuutit. Kuvasto nähdään, että Imagen tyypiksi on valittu Post Image ja Attributesissa on ”All Attributes”. Name ja Entity Alias ovat ”ExamplePluginPost”. Toinenkin image rekisteröityi oikein.

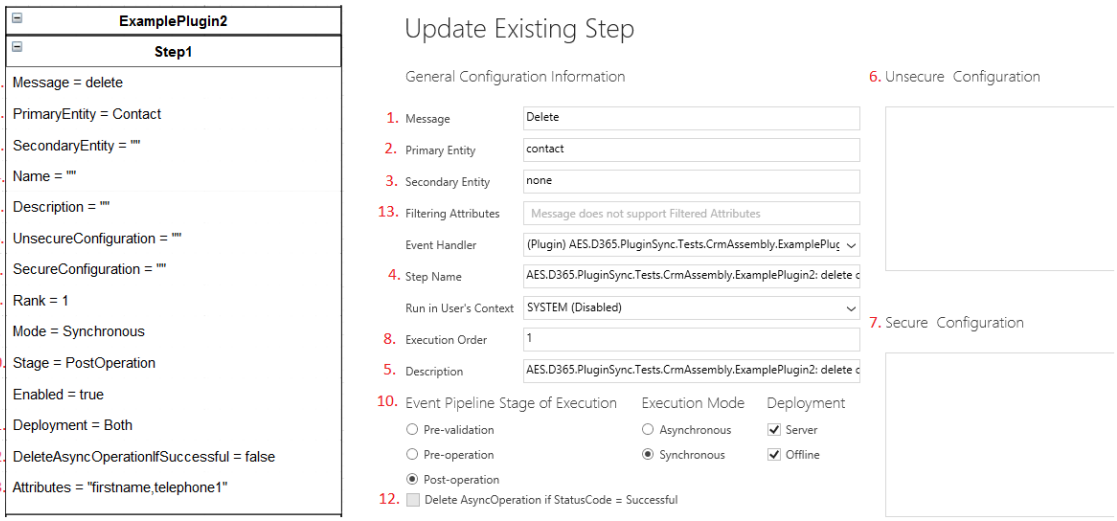
Seuraavaksi tarkistettiin ensimmäisen pluginin toinen steppi. Sen konfiguraatio on kuvassa 21.



KUVA 21. Ensimmäisen pluginin toisen stepin konfiguraatio

Ensimmäisen pluginin toinen steppi oli halutulta konfiguraatioltaan erilainen kuin ensimmäinen steppi. Kuvasta 21 huomataan, että haluttu konfiguraatio on saman kuin toteutunut konfiguraatio. Stepillä ei ollut imageja. Ensimmäisen pluginin molemmat stepit ovat siis rekisteröityneet onnistuneesti.

Ensimmäinen plugini rekisteröityi täysin ohjeiden mukaan. Jotta saataisiin varmuutta ohjelman toimimisesta, varmistettiin toisenkin pluginin rekisteröinti. Toisella pluginilla oli vain yksi steppi, jonka konfiguraatio on kuvassa 22.

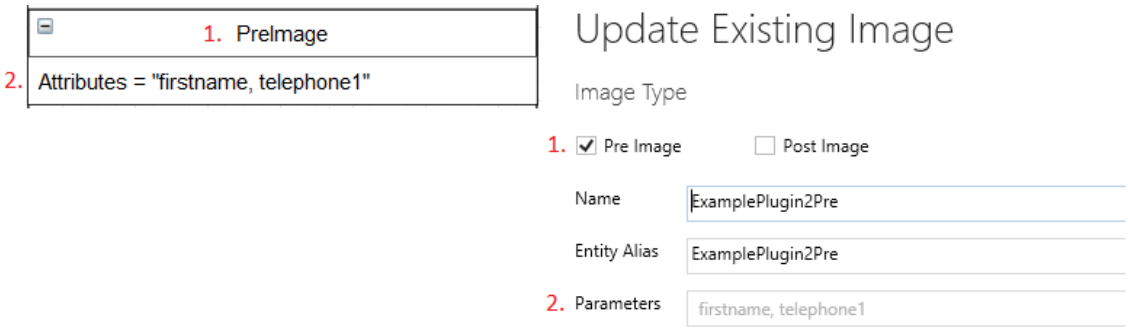


KUVA 22. Toisen pluginin steppi

Kuvan 22 perusteella todettiin, että steppi on rekisteröitynyt oikein. Halutut arvot ovat samat kuin toteutuneet arvot. Stepin nimi ja kuvauskin ovat muodostuneet oikein muotoon ”AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin2 : delete of

contact”. Ainoana erona on kuvassa numeroidut kohdat 13. Haluttuun konfiguraatioon on valittu attribuuteiksi firstname ja telephone1. Steppi on rekisteröity delete-messageen, joka ei tue attributes-parametrin asettamista. Se näkyy toteutuneessa rekisteröinnissä viestinä ”Message does not support Filtered Attributes”. Tämä johtuu siitä, että Delete-viesti vastaa koko tietueen poistoa. Jos tietueesta tyhjennettäisiin yksi kenttä, olisi se silloin Update-viesti.

Toisen pluginin stepillä oli myös yksi image. Myös sen konfiguraatio tarkistettiin. Imagen konfiguraatio on kuvassa 23.



KUVA 23. Toisen pluginin stepin imagen konfiguraatio

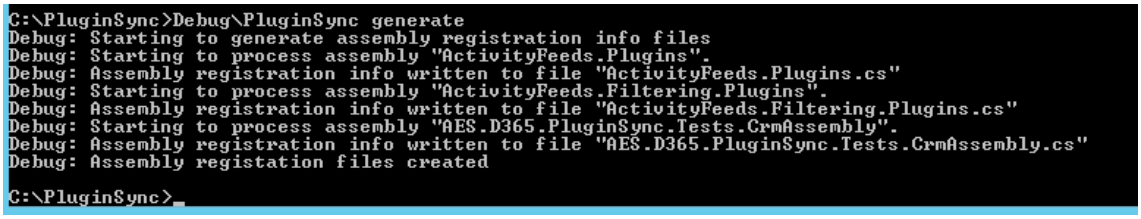
Kuvan 23 perusteella imagen tyypiksi oli haluttu Pre Image ja parametreiksi firstname ja telephone1. Nämä valinnat ovat myös toteutuneella imagella. Nimeksi ja Entity Aliakseksi on tullut ”ExamplePlugin2Pre”, eli myös nimeäminen on toiminut oikein. Voidaan siis todeta imagen rekisteröityneen oikein.

Testauksen perusteella voidaan todeta, että rekisteröinti toimii ainakin testin tapauksessa. Testitapauksen perusteella ei voida sanoa, että rekisteröinti toimii virheettä kaikissa tapauksissa. Testitapauksen oli tehty niin, että ne vastaavat oikeanlaista konfiguraatiota ja sen tarkoitus oli todentaa, että suotuisilla syötteillä se pystyy rekisteröimään plugineja CRM-ympäristöön. Testeissä ohjelma oli rekisteröinyt pluginit annettujen ohjeiden mukaisesti, joten rekisteröinnin voidaan sanoa toimivan.

5.2 Rekisteröintiohjeiden generoiminen

Rekisteröintiohjeiden lähtötilanne oli se, mihin luvussa 6.1 jäätiin. Organisaatiossa oli rekisteröitynä neljä assemblyä, joista 3 oli oletusassemblyjä ja yksi oli ulkoisesti

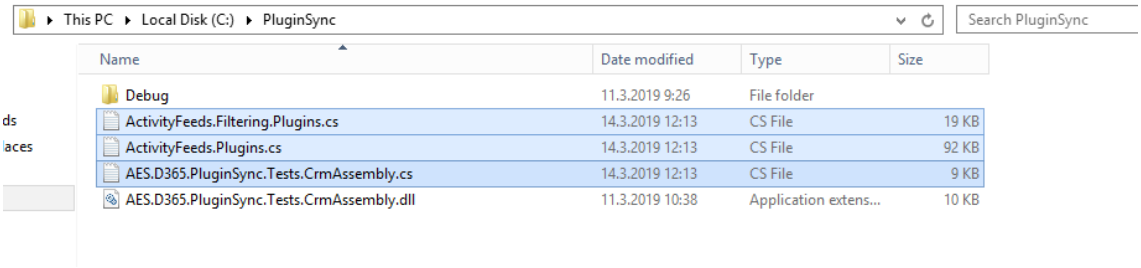
rekisteröity. Ulkoisesti rekisteröity assembly oli kuvan 14 mukainen. Pluginsync suoritettiin komennolla ”PluginSync generate”. Ajotulos on kuvassa 24.



KUVA 24. Rekisteröintiohjeiden luonnin testiajo

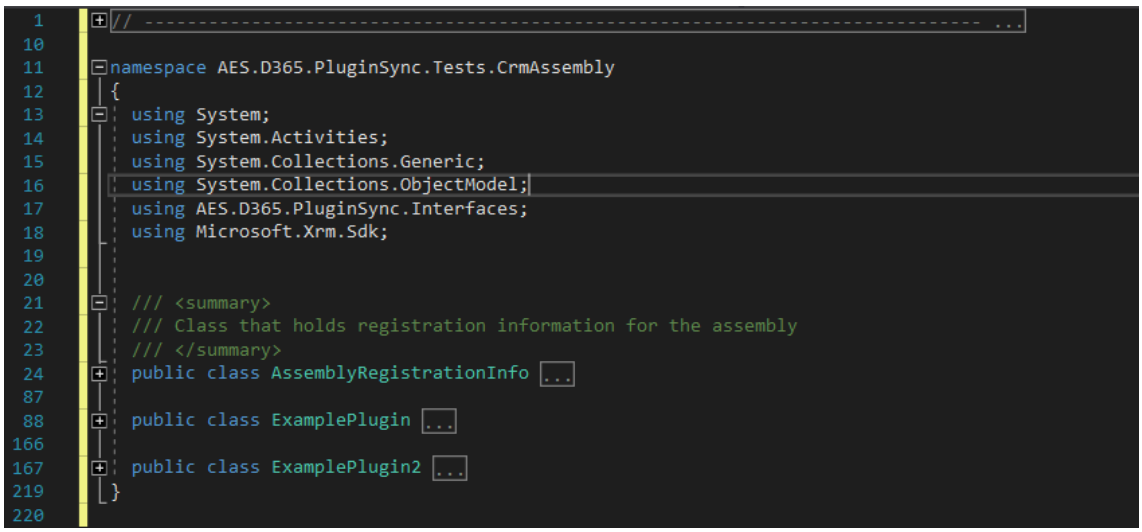
Kuvasta 24 nähdään, että ohjelma on prosessoinut organisaatiossa olevat assemblyt vuorotellen. Plugin Registration Toolilla organisaatiossa näkyi olevan neljä assemblyä, mutta PluginSync loi ohjeet vain kolmelle assemblylle. Tämä johtuu siitä, että tietyistä oletusassemblyistä ei voi ohjelmallisesti hakea tietoa. Ohjeet luodaan siis vain niistä assemblyistä, jotka PluginSync näkee.

Ohjeiden luonnissa ei ole mahdollisuutta, että ohjeet luotaisiin vain valituista assemblyistä. Luodut ohjeet tallentuvat kansioon, josta ohjelma suoritettiin. Testiajossa ohjelma ajettiin kansiosta C:\PluginSync, johon ohjeet luotiin. Resurssienhallinnan avulla tarkistetiin, oliko ohjeet luotu. Kansion sisältö on kuvassa 25



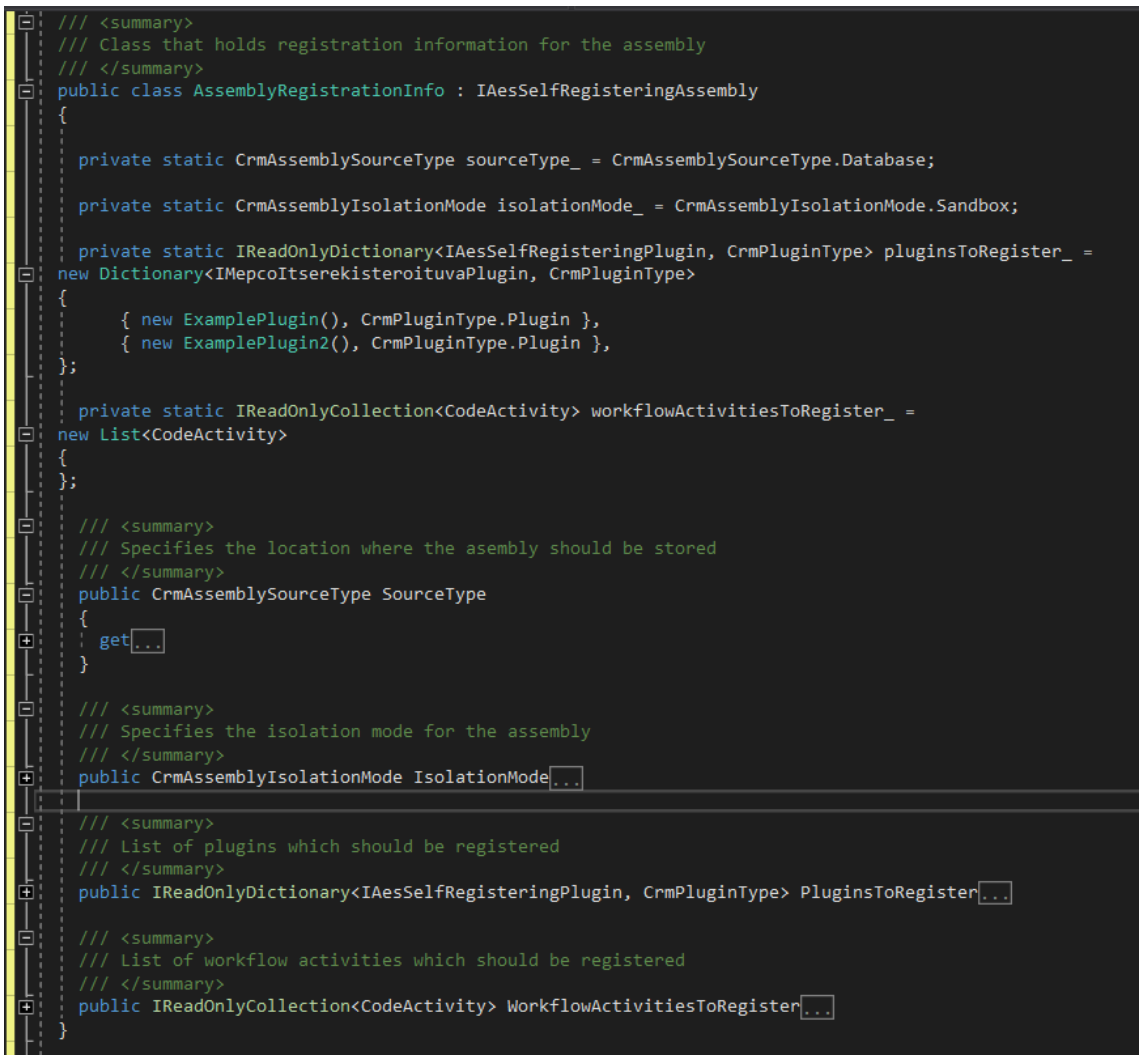
KUVA 25. Luodut ohjetiedostot

Kuvasta 25 todetaan, että pluginien rekisteröintiohjeet ovat muodostuneet. ”ActivityFeeds”-alkuiset tiedostot ovat rekisteröintiohjeita oletusassemblyille, joten ne eivät ole merkityksellisiä. Rekisteröintiohjeiden oikeellisuus tarkistettiin vertaamalla luotua kooditiedostoa organisaatiossa olevaan rekisteröintiin. Kuvassa 26 on tiedoston sisällön ylätasot.



KUVA 26. Rekisteröintiohjeet

Kuvassa 26 näkyy PluginSyncin luoman rekisteröintiohjeen luokat. Tiedostoon on luotu nimiavaruus nimeltä ”AES.D365.PluginSync.Tests.CrmAssembly”. Nimiavaruus on CRM:ssä olevan assemblyn nimi. Nimiavaruuden jälkeen tiedostossa on käytetyt ulkoiset kirjastot. Käytettyjen kirjastojen jälkeen on luokka, joka sisältää assemblytason rekisteröintiohjeet. Sen jälkeen on vielä omissa luokissaan rekisteröintiohjeet jokaiselle assemblyssä olevalle pluginille. Luotu rekisteröintiohje on liitteenä 1.



```
/// <summary>
/// Class that holds registration information for the assembly
/// </summary>
public class AssemblyRegistrationInfo : IAesSelfRegisteringAssembly
{

 private static CrmAssemblySourceType sourceType_ = CrmAssemblySourceType.Database;

 private static CrmAssemblyIsolationMode isolationMode_ = CrmAssemblyIsolationMode.Sandbox;

 private static IReadOnlyDictionary<IAesSelfRegisteringPlugin, CrmPluginType> pluginsToRegister_ =
new Dictionary<IMepcoItserekisteroituvaPlugin, CrmPluginType>
{
     { new ExamplePlugin(), CrmPluginType.Plugin },
     { new ExamplePlugin2(), CrmPluginType.Plugin },
};

 private static IReadOnlyCollection<CodeActivity> workflowActivitiesToRegister_ =
new List<CodeActivity>
{
};

 /// <summary>
 /// Specifies the location where the asembly should be stored
 /// </summary>
 public CrmAssemblySourceType SourceType
 {
  get...
 }

 /// <summary>
 /// Specifies the isolation mode for the assembly
 /// </summary>
 public CrmAssemblyIsolationMode IsolationMode...

 /// <summary>
 /// List of plugins which should be registered
 /// </summary>
 public IReadOnlyDictionary<IAesSelfRegisteringPlugin, CrmPluginType> PluginsToRegister...

 /// <summary>
 /// List of workflow activities which should be registered
 /// </summary>
 public IReadOnlyCollection<CodeActivity> WorkflowActivitiesToRegister...
}
```

KUVA 27. Assemblylle luotu rekisteröintiohje

Kuvassa 27 on assemblyn rekisteröintiohjeet. Assembly, jolle ohjeet luotiin, on kuvan 15 mukainen. Luoduissa ohjeissa assemblyn Sourcetypeksi on asetettu Database ja IsolationModeksi Sandbox. Nämä asetukset ovat kuten pitääkin. Assemblyn sisältämät pluginit ovat listattuna muuttujassa pluginsToRegister_. Sen sisältönä on instanssit luokista ExamplePlugin ja ExamplePlugin2. Pluginit ovat myös kuvan 14 mukaisesti.

Seuraavaksi tarkasteltiin plugineille luotuja rekisteröintiohjeita. Kuvasta 27 nähdään, että molemmille assemblyssä oleville plugineille luotiin omat luokkansa. Luokkien sisältönä on pluginien rekisteröintiohjeet. Examplepluginin rekisteröintiohjeet ovat kuvassa 28.

```
public class ExamplePlugin : IPlugin, IAesSelfRegisteringPlugin
{

  /// <summary>
  /// Registration instructions of all steps of this plugin
  /// </summary>
  public IReadOnlyCollection<IAesSelfRegisteringStep> PluginSteps
  {
    get
    {
      return GetPluginSteps();
    }
  }

  private IReadOnlyCollection<IAesSelfRegisteringStep> GetPluginSteps()...
}
```

KUVA 28. Examplepluginin rekisteröintiohjeet

Kuvasta 28 huomataan, että pluginille luotiin kaksi metodia. Toinen on julkinen PluginsSteps, joka palauttaa pluginin stepit. Steppien palautus tehdään kutsumalla yksityistä metodia GetPluginSteps, jossa on määritelty steppien asetukset. Kuvassa 29 on Examplepluginin ensimmäisen stepin asetukset.

```
List<IAesSelfRegisteringStep> returnValue = new List<IAesSelfRegisteringStep>();
var step1 = new AesSelfRegisteringStep();
step1.Message = "Update";
step1.PrimaryEntity = "account";
step1.SecondaryEntity = "none";
step1.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: update of account";
step1.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: update of account";
step1.UnsecureConfiguration = "test123";
step1.SecureConfiguration = "";
step1.Rank = 1;
step1.Mode = CrmPluginStepMode.Synchronous;
step1.Stage = CrmPluginStepStage.PostOperation;
step1.Enabled = true;
step1.Deployment = CrmPluginStepDeployment.Both;
step1.DeleteAsyncOperationIfSuccessful = false;
step1.UserName = "";
step1.Attributes = new Collection<string>()
{
    "name",
    "telephone1",
    "accountnumber",
    "numberofemployees",
};
var step1PreImage = new AesSelfRegisteringImage();
step1PreImage.ImageType = CrmPluginImageType.PreImage;
step1PreImage.Attributes = new Collection<string>()
{
    "name",
    "telephone1",
};
var step1PostImage = new AesSelfRegisteringImage();
step1PostImage.ImageType = CrmPluginImageType.PostImage;
step1PostImage.Attributes = new Collection<string>()
{
};
step1.Images = new Collection<IAesSelfRegisteringImage>
{
    step1PreImage,step1PostImage
};
returnValue.Add(step1);
```

KUVA 29. Examplepluginin stepin 1 luodut ohjeet

GetPluginSteps-metodin ensimmäinen käsky on luoda lista, jonka sisällöksi myöhemmin laitetaan luodut stepit. Listan luomisen jälkeen on stepin 1 asetukset. Kun asetuksia vertaa kuvaan 15, huomataan, että asetukset pääosin vastaavat toisiaan. Nimi ja kuvaus ovat eri, sillä ne ovat PluginSyncin luomia oletustekstejä, jos käyttäjä ei rekisteröitäessä määrittänyt nimeksi ja kuvaukseksi mitään. Myös stepin SecureConfiguration on tyhjä, vaikka kuvassa 15 sen arvona oli ”test456”. Tämä johtuu siitä, että SecureConfiguration on nimensä mukaisesti turvallinen lisäkoniguraatio, eikä sitä voi ohjelmallisesti hakea. Käyttäjän on itse osattava lisätä ohjeisiin tuo arvo, mikäli se on tarpeellinen. Yleensä sen arvo on kuitenkin tyhjä.

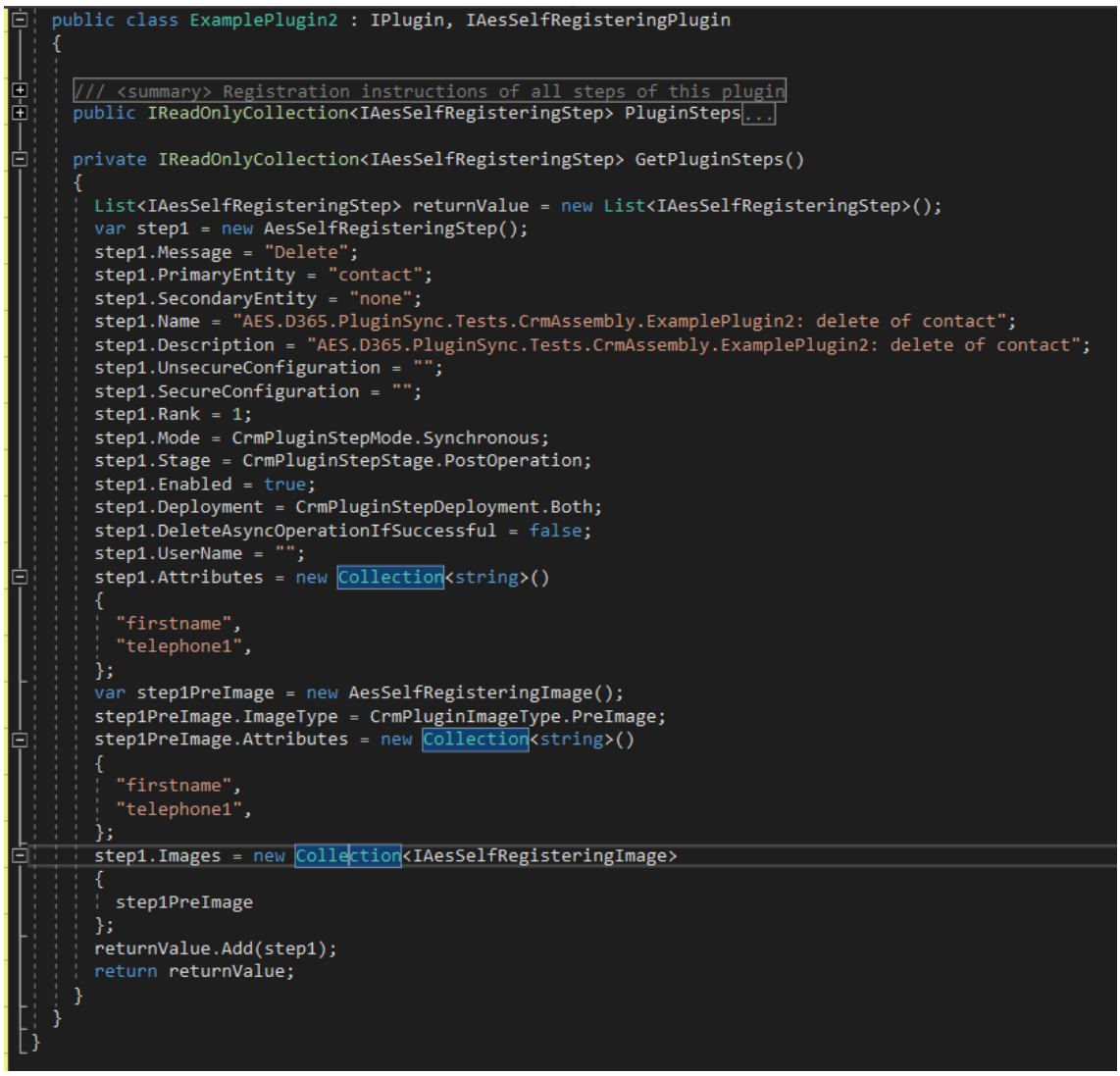
Stepille 1 luotiin rekisteröintiohjeet oikein. Myös stepille 2 luodut ohjeet tarkistettiin. Ohjeet ovat kuvassa 30.

```
var step2 = new AesSelfRegisteringStep();
step2.Message = "Create";
step2.PrimaryEntity = "account";
step2.SecondaryEntity = "none";
step2.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: create of account";
step2.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: create of account";
step2.UnsecureConfiguration = "";
step2.SecureConfiguration = "";
step2.Rank = 2;
step2.Mode = CrmPluginStepMode.Asynchronous;
step2.Stage = CrmPluginStepStage.PostOperation;
step2.Enabled = true;
step2.Deployment = CrmPluginStepDeployment.ServerOnly;
step2.DeleteAsyncOperationIfSuccessful = true;
step2.UserName = "";
step2.Attributes = new Collection<string>()
{
};
returnValue.Add(step2);
return returnValue;
```

KUVA 30. Stepin 2 ohjeet

Stepille 2 luodut ohjeet ovat huomattavasti lyhyemmät kuin stepille 1. Pääosin tse johtuu siitä, että stepillä 2 ei ole imageja. Siitä huolimatta nämä ohjeet ovat aivan yhtä validit. Kuvaan 15 verrattaessa asetukset ovat pääosin oikein. Kuten stepillä 1, on stepin 2 nimi ja kuvaus eri kuin kuvassa 15. Se johtuu tässäkin tapauksessa siitä, että assembly reksiteröintiin PluginSyncillä, ja se loi assemblylle nimen ja kuvauksen, koska niitä ei alkuperäisessä ohjeessa oltu määritelty. Koodin lopussa on käsky, jolle steppi 2 lisätään palautettavien steppien listaan. sen jälkeen on käsky ”return returnValue;”, joka palauttaa listan stepeistä.

Exampluginille oli luotu ohjeet oikein. Assemblyssä oli toinenkin plugin nimeltä Exampleplugin2. Sille luodut rekisteröintiohjeet tarkistettiin myös. Luodut ohjeet ovat kuvassa 31.



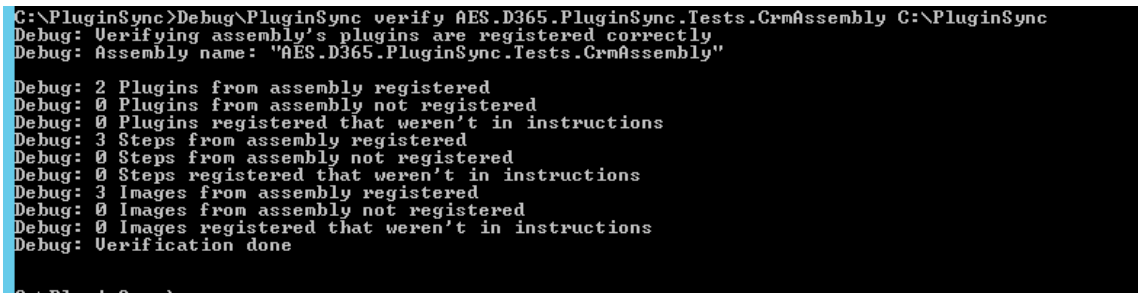
```
public class ExamplePlugin2 : IPlugin, IAesSelfRegisteringPlugin
{

  /// <summary> Registration instructions of all steps of this plugin
  public IReadOnlyCollection<IAesSelfRegisteringStep> PluginSteps...

  private IReadOnlyCollection<IAesSelfRegisteringStep> GetPluginSteps()
  {
    List<IAesSelfRegisteringStep> returnValue = new List<IAesSelfRegisteringStep>();
    var step1 = new AesSelfRegisteringStep();
    step1.Message = "Delete";
    step1.PrimaryEntity = "contact";
    step1.SecondaryEntity = "none";
    step1.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin2: delete of contact";
    step1.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin2: delete of contact";
    step1.UnsecureConfiguration = "";
    step1.SecureConfiguration = "";
    step1.Rank = 1;
    step1.Mode = CrmPluginStepMode.Synchronous;
    step1.Stage = CrmPluginStepStage.PostOperation;
    step1.Enabled = true;
    step1.Deployment = CrmPluginStepDeployment.Both;
    step1.DeleteAsyncOperationIfSuccessful = false;
    step1.UserName = "";
    step1.Attributes = new Collection<string>()
    {
      "firstname",
      "telephone1",
    };
    var step1PreImage = new AesSelfRegisteringImage();
    step1PreImage.ImageType = CrmPluginImageType.PreImage;
    step1PreImage.Attributes = new Collection<string>()
    {
      "firstname",
      "telephone1",
    };
    step1.Images = new Collection<IAesSelfRegisteringImage>
    {
      step1PreImage
    };
    returnValue.Add(step1);
    return returnValue;
  }
}
```

KUVA 31. Exampleplugin2:lle luodut ohjeet

Kuvan 31 perusteella todetaan, että pluginille on luotu ohjeet oikein. Luokassa on samat metodit kuin Exampleplugin illakin. Tällä pluginilla oli vain yksi steppi, jonka asetukset on määritelty metodissa GetPluginSteps. Vertaamalla luotuja ohjeita kuvaan 14, voidaan luotujen ohjeiden todeta vastaavan lähtötilannetta.

5.3 Verifiointi

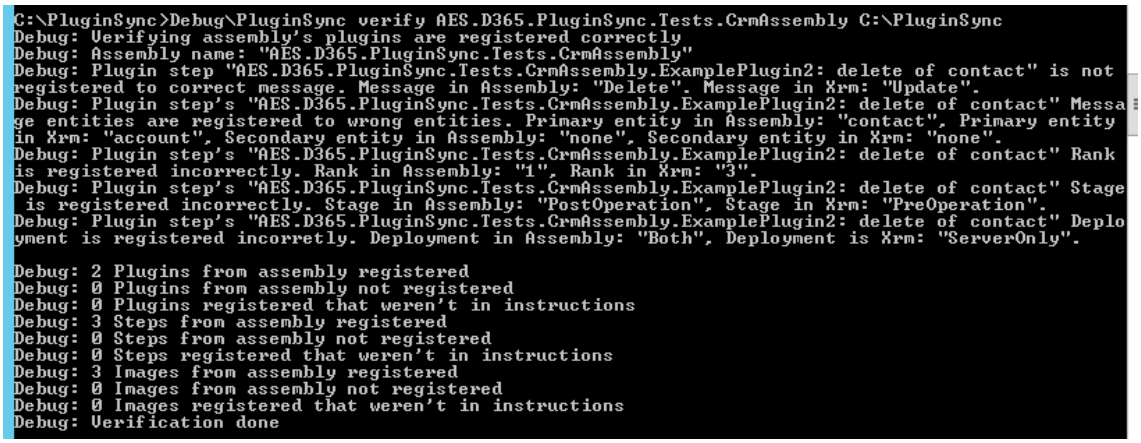
Ohjeiden verifiointia testattiin tilanteessa, jossa CRM:ään oli rekisteröity kuvan 14 mukainen assembly. Ohjelma ajettiin komennolla ”Pluginsync verify AES.D365.PluginSync.Tests.CrmAssembly C:\PluginSync”. Ajotulos on kuvassa 33.



KUVA 32. Assemblyn verifiointi

Kuvassa näkyy assemblyn verifioinnin tulos. Ajon alussa oli informaatioviestejä, joissa ilmoitettiin ajon käynnistyneen. Sen jälkeen komentoriville tulostui lista eroista valitun assemblyn ja CRM:stä löytyneen assemblyn välillä. Ensin listataan erot pluginien välillä, sitten steppien välillä ja viimeisenä imageiden välillä.

Kuvan 33 testiajossa valitun assemblyn ja CRM:stä löytyneen assemblyn välillä ei ollut eroja. Kaikki assemblystä löytyneet pluginit oli rekisteröity ohjeiden mukaisesti eikä ylimääräisiä pluginejakaan ollut. Seuraavaksi testattiin tilanne, jossa reksiteröinti ei vastannut verifioitavan assemblyn ohjeita. Tilanteen ajotulos on kuvassa 33.



KUVA 33. Assemblyn verifiointi, kun rekisteröinti ei täsmää

Kuvasta 33 huomataan, että nyt komentoriville tulostui enemmän tekstiä. Tämä johtuu siitä, että assemblyn väärin konfiguroidut asetukset tulostetaan komentoriville. Testitapausta varten oli muutettu Exampleplugin2:n stepin asetuksia. Pluginsync tunnisti muutetut asetukset. Väärät asetukset tulostettiin, jotta käyttäjä voi halutessaan korjata virheellisen konfiguroinnin.

Kuvan 33 testitapauksessa komentorivin loppuun tulostunut lista on edelleen saman lainen kuin kuvassa 32. Tämä johtuu siitä, että listaan tulostetaan eroavaisuudet pluginien,

steppien ja imageiden lukumäärissä. Testitapaukseen ei oltu muuteltu asioiden määrää, vaan vain muutettu yhden stepin konfiguraatiota. Jos esimerkiksi toinen plugin olisi positettu kokonaan, olisi tieto siitä tulostunut listaan.

6 POHDINTA

Työssä toteutettiin konsolisovellus, jolla pystyy rekisteröimään plugineja CRM-organisaatioihin. Ohjelman toinen toiminto on luoda rekisteröintiohjeet plugineille, jotka on jo rekisteröity CRM:ään. Ohjelmalla pystyy myös verifioimaan, että CRM:ään rekisteröidyt pluginit on rekisteröity oikein. Ohjelman toteutettiin niin, että sen voi suorittaa skriptin avulla.

Ohjelman tärkein toiminnallisuus on pluginien rekisteröinti. Skriptien avulla rekisteröinti voidaan automatisoida esimerkiksi koodin kääntämisen yhteyteen. Tällä tavoin pluginia kehitettäessä saadaan testausympäristöön aina uusin versio automaattisesti. Se nopeuttaisi ja helpottaisi kehitystyötä, sillä käsin rekisteröinnissä kuluu aikaa ja siinä voi tapahtua helposti virheitä. Lisäksi kehittäjän ei täytyisi erikseen muistaa tehdä rekisteröintiä, koska se hoituisi aina kääntämisen yhteydessä.

Toteutettu ohjelma pystyy suoriutumaan onnistuneesti sille tehdyistä toiminnoista. Ohjelman testaaminen oli kuitenkin hyvin pintapuolista. Testitapaukset olivat tilanteita, joissa muuttujat olivat mahdollisimman suotuisat. Testeillä haluttiin osoittaa, että ohjelma voi hyvissä olosuhteissa toimia oikein. Se on vastoin testaamisen yleistä periaatetta, joka on osoittaa, että ohjelma ei toimi oikein. Esimerkiksi rekisteröinti huonoilla ohjeilla tai väärät yhteystiedot CRM:ään voivat aiheuttaa virheitä, joihin ohjelma ei ole varautunut. Ohjelman kattava testaaminen ja virheisiin varautuminen ovat jatkokehitysasioita.

Muitakin jatkokehitysideoita on. Ensimmäinen liittyy pluginien rekisteröintiin. Pluginien rekisteröinti toteutettiin niin, että ohjelmalle syötetetään parametrinä assemblyn tarkka nimi ja sijainti. Yhdellä ajolla voi siis rekisteröidä vain yhden assemblyn. Jos assemblyja on monia, täytyy jokaiselle tehdä erikseen oma ajonsa. Ohjelmaa voisi muuttaa niin, että sille voisi antaa parametrinä kansion sijainnin ja ohjelma osaisi hakea itse kansiosta kaikki assemblyt ja rekisteröidä ne. Tällöin yhdellä ajokerralla rekisteröitäisiin monta assemblya, joka on parempi ratkaisu kuin tehdä erillinen ajo jokaiselle erikseen.

Toinen jatkokehitysidea liittyy rekisteröintiohjeiden luomiseen. Ohjelma toteutettiin niin, että se hakee organisaatiosta kaikki muokattavissa olevat assemblyt ja tekee jokaiselle rekisteröintiohjeet. Jos organisaatiossa on paljon assemblyjä, luodaan paljon turhia rekisteröintiohjeita. Ohjelmaan olisi hyvä toteuttaa suodatin, jolla voisi rajoittaa, mistä

assemblyistä ohjeet luodaan. Esimerkiksi Accountor Enterprise Solutions nimeää assemblyt niin, että niiden nimi alkaa kirjaimilla ”AES”. Ohjelmaan voisi lisätä suodattimen, joka ottaa parametrinä halutun tekstijonon ja tekee ohjeet vain assemblyistä, joista annettu tekstijono löytyy. Tällöin ohjeet luotaisiin vain halutuista assemblyistä, eikä turhia tiedostoja syntyisi niin paljoa.

LÄHTEET

365 Blog. 2017. What's new for Customer Engagement developer documentation in version 9.0. Julkaistu 1.11.2017. Luettu 15.11.2018. https://blogs.msdn.microsoft.com/crm/2017/11/01/whats-new-for-customer-engagement-developer-documentation-in-version-9-0/.

Build Numbers. N.d. Luettu 10.10.2018. https://buildnumbers.wordpress.com/crm/.

CRM Book. N.d. When to use plugins. Luettu 10.10.2018. https://crmbook.powerobjects.com/extending-crm/plug-in-development-and-workflow-extensions/plug-ins/when-to-use-plug-ins/.

Koskimies, K & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Helsinki: Talentum.

Microsoft. 2016. IPlugin.Execute Method. Julkaistu 28.11.2016. Luettu 28.11.2018. https://docs.microsoft.com/en-us/previous-versions/dynamicscrm-2016/developers-guide/gg326167%28v%3dcrm.8%29.

Microsoft. 2017a. Customize your Dynamics 365 system. Päivitetty 28.11.2016. Luettu 10.10.2018. https://technet.microsoft.com/en-us/library/dn531158.aspx.

Microsoft, 2017b. On-premise plug-in development. Julkaistu 31.10.2017. Luettu 10.10.2018. https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/plugin-development.

Microsoft. 2017c. Register a plug-in to be deployed on-premise. Julkaistu 31.10.2018. Luettu 10.10.2018. https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/register-deploy-plugins#plug-in-registration-tool.

Microsoft. 2018. Tutorial: Write and register a plug-in. Julkaistu 31.10.2018. Luetty 19.3.2019. https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/tutorial-write-plug-in

Microsoft. 2019. TFS. Luettu 20.2.2019. https://visualstudio.microsoft.com/tfs/

Microsoft. N.d. IPlugin Interface. Luettu 11.10.2018. https://docs.microsoft.com/en-us/dotnet/api/microsoft.xrm.sdk.iplugin?view=dynamics-general-ce-9

Molag, T. 2016. What is Dynamics 365?. Julkaistu 6.7.2016. Päivitetty 15.11.2018. Luettu. 10.10.2018. https://www.encorebusiness.com/blog/what-is-microsoft-dynamics-365/.

LIITTEET

Liite 1. Generoitu rekisteröintiohjetiedosto

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AES.D365.PluginSync.Tests.CrmAssembly
{
    using System;
    using System.Activities;
    using System.Collections.Generic;
    using System.Collections.ObjectModel;
    using AES.D365.PluginSync.Interfaces;
    using Microsoft.Xrm.Sdk;


    /// <summary>
    /// Class that holds registration information for the assembly
    /// </summary>
    public class AssemblyRegistrationInfo : IAesSelfRegisteringAssembly
    {

        private static CrmAssemblySourceType sourceType_ = CrmAssemblySourceType.Database;

        private static CrmAssemblyIsolationMode isolationMode_ = CrmAssemblyIsolationMode.Sandbox;

        private static IReadOnlyDictionary<IAesSelfRegisteringPlugin, CrmPluginType> pluginsToRegister_ =
            new Dictionary<IMepcoItserekisteroituvaPlugin, CrmPluginType>
            {
                { new ExamplePlugin(), CrmPluginType.Plugin },
                { new ExamplePlugin2(), CrmPluginType.Plugin },
            };

        private static IReadOnlyCollection<CodeActivity> workflowActivitiesToRegister_ =
            new List<CodeActivity>
            {
            };

        /// <summary>
        /// Specifies the location where the asembly should be stored
        /// </summary>
        public CrmAssemblySourceType SourceType
        {
            get
            {
                return sourceType_;
            }
        }

        /// <summary>
        /// Specifies the isolation mode for the assembly
        /// </summary>
        public CrmAssemblyIsolationMode IsolationMode
        {
            get
            {
                return isolationMode_;
            }
        }

        /// <summary>
        /// List of plugins which should be registered
        /// </summary>
        public IReadOnlyDictionary<IAesSelfRegisteringPlugin, CrmPluginType> PluginsToRegister
        {
            get
```

```
        {
            return pluginsToRegister_;
        }
    }

    /// <summary>
    /// List of workflow activities which should be registered
    /// </summary>
    public IReadOnlyCollection<CodeActivity> WorkflowActivitiesToRegister
    {
        get
        {
            return workflowActivitiesToRegister_;
        }
    }
}

public class ExamplePlugin : IPlugin, IAesSelfRegisteringPlugin
{

    /// <summary>
    /// Registration instructions of all steps of this plugin
    /// </summary>
    public IReadOnlyCollection<IAesSelfRegisteringStep> PluginSteps
    {
        get
        {
            return GetPluginSteps();
        }
    }

    private IReadOnlyCollection<IAesSelfRegisteringStep> GetPluginSteps()
    {
        List<IAesSelfRegisteringStep> returnValue = new List<IAesSelfRegisteringStep>();
        var step1 = new AesSelfRegisteringStep();
        step1.Message = "Update";
        step1.PrimaryEntity = "account";
        step1.SecondaryEntity = "none";
        step1.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: update of account";
        step1.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: update of account";
        step1.UnsecureConfiguration = "test123";
        step1.SecureConfiguration = "";
        step1.Rank = 1;
        step1.Mode = CrmPluginStepMode.Synchronous;
        step1.Stage = CrmPluginStepStage.PostOperation;
        step1.Enabled = true;
        step1.Deployment = CrmPluginStepDeployment.Both;
        step1.DeleteAsyncOperationIfSuccessful = false;
        step1.UserName = "";
        step1.Attributes = new Collection<string>()
        {
            "name",
            "telephone1",
            "accountnumber",
            "numberofemployees",
        };
        var step1PreImage = new AesSelfRegisteringImage();
        step1PreImage.ImageType = CrmPluginImageType.PreImage;
        step1PreImage.Attributes = new Collection<string>()
        {
            "name",
            "telephone1",
        };
        var step1PostImage = new AesSelfRegisteringImage();
        step1PostImage.ImageType = CrmPluginImageType.PostImage;
        step1PostImage.Attributes = new Collection<string>()
        {
        };
        step1.Images = new Collection<IAesSelfRegisteringImage>
        {
            step1PreImage,step1PostImage
        };
        returnValue.Add(step1);
        var step2 = new AesSelfRegisteringStep();
        step2.Message = "Create";
        step2.PrimaryEntity = "account";
        step2.SecondaryEntity = "none";
        step2.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: create of account";
        step2.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin: create of account";
```

```
        step2.UnsecureConfiguration = "";
        step2.SecureConfiguration = "";
        step2.Rank = 2;
        step2.Mode = CrmPluginStepMode.Asynchronous;
        step2.Stage = CrmPluginStepStage.PostOperation;
        step2.Enabled = true;
        step2.Deployment = CrmPluginStepDeployment.ServerOnly;
        step2.DeleteAsyncOperationIfSuccessful = true;
        step2.UserName = "";
        step2.Attributes = new Collection<string>()
        {
        };
        returnValue.Add(step2);
        return returnValue;
    }
}

public class ExamplePlugin2 : IPlugin, IAesSelfRegisteringPlugin
{

    /// <summary>
    /// Registration instructions of all steps of this plugin
    /// </summary>
    public IReadOnlyCollection<IAesSelfRegisteringStep> PluginSteps
    {
        get
        {
            return GetPluginSteps();
        }
    }

    private IReadOnlyCollection<IAesSelfRegisteringStep> GetPluginSteps()
    {
        List<IAesSelfRegisteringStep> returnValue = new List<IAesSelfRegisteringStep>();
        var step1 = new AesSelfRegisteringStep();
        step1.Message = "Delete";
        step1.PrimaryEntity = "contact";
        step1.SecondaryEntity = "none";
        step1.Name = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin2: delete of contact";
        step1.Description = "AES.D365.PluginSync.Tests.CrmAssembly.ExamplePlugin2: delete of contact";
        step1.UnsecureConfiguration = "";
        step1.SecureConfiguration = "";
        step1.Rank = 1;
        step1.Mode = CrmPluginStepMode.Synchronous;
        step1.Stage = CrmPluginStepStage.PostOperation;
        step1.Enabled = true;
        step1.Deployment = CrmPluginStepDeployment.Both;
        step1.DeleteAsyncOperationIfSuccessful = false;
        step1.UserName = "";
        step1.Attributes = new Collection<string>()
        {
            "firstname",
            "telephone1",
        };
        var step1PreImage = new AesSelfRegisteringImage();
        step1PreImage.ImageType = CrmPluginImageType.PreImage;
        step1PreImage.Attributes = new Collection<string>()
        {
            "firstname",
            "telephone1",
        };
        step1.Images = new Collection<IAesSelfRegisteringImage>
        {
            step1PreImage
        };
        returnValue.Add(step1);
        return returnValue;
    }
```