ADEventService – integration af IT-fagsystem til Active Directory (AD)

Revision			
Dato/Initialer	Handling	Se også	QA/Dato/Initialer
20150804/SDE	Diverse korrekturrettelser		
20140106/SDE	Tilføjet baggrundsafsnit		
20130902/SDE	Første udkast		

1 Indledning

Dette notat beskriver, hvorledes et IT-fagsystem integreres med GKs Active Directory (AD) ved hjælp af integrations motoren **ADEventService**.

Spørgsmål eller kommentarer til dokumentet her, eller til ADEventServicen kan stiles til GKs Drift og Arkitektur enhed (ITA).

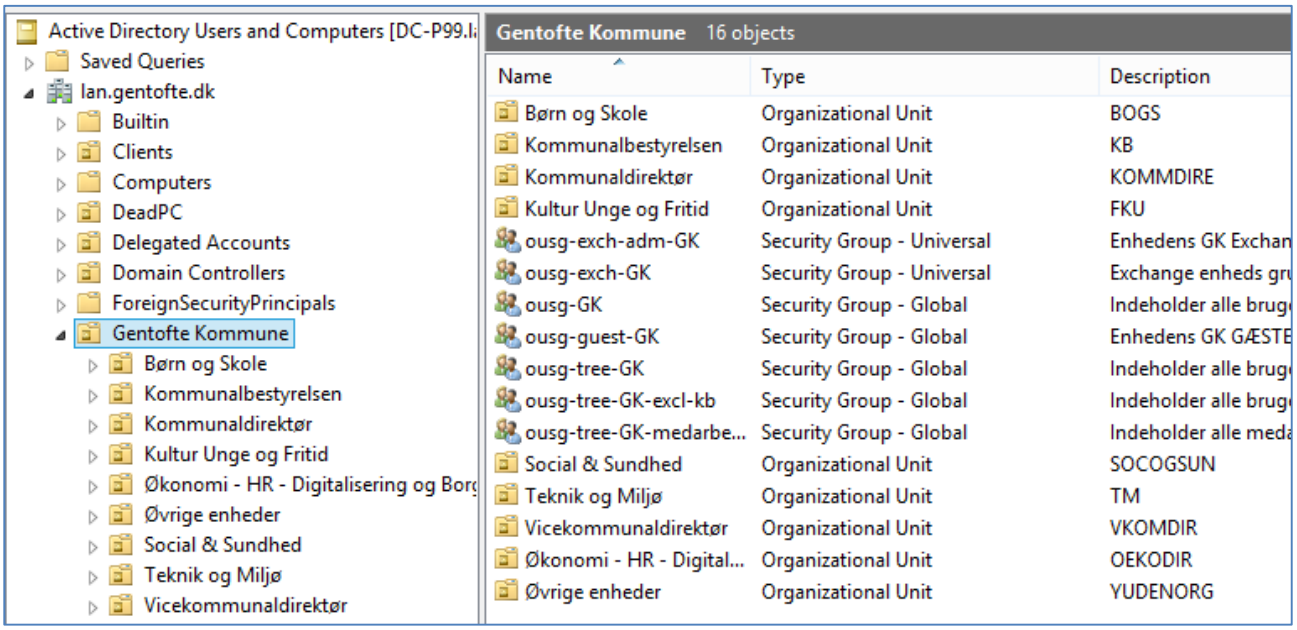
2 Baggrund

Microsoft Active Directory (AD) er for mange virksomheder en central infrastruktur komponent for administration af virksomhedens IT-brugere, IT-grupper (grupper af brugere, PC'ere mv.), organisatoriske enheder og andre IT-nære ressourcer. Herunder benyttes AD også til at vedligeholde IT-brugernes autorisationer til IT fagsystemer og andre IT-ressourcer via ADs koncept for sikkerhedsgrupper.

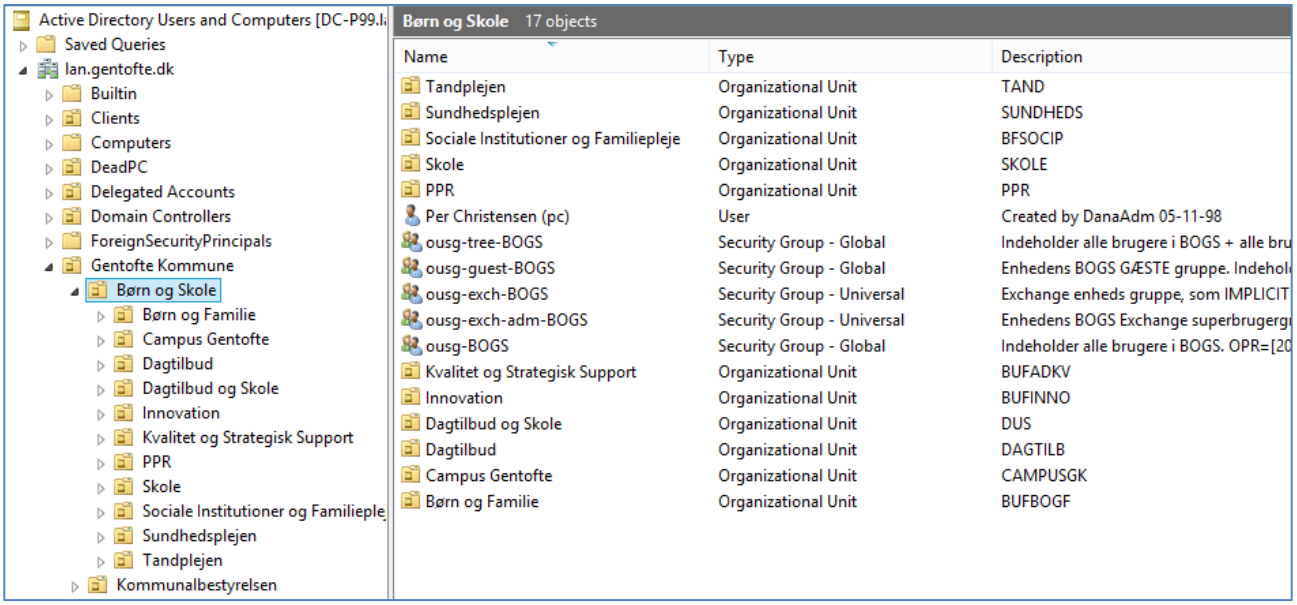
Det er også tilfældet i Gentofte Kommune (GK), som med omkring 6500 brugere, 500 organisatoriske enheder og ca. 300+ fagsystemer anvender AD i udstrakt grad til at vedligeholde stamoplysninger omkring disse entiteter.

I GK er det AD der sidder i førersædet (*såkaldt autoritativ kilde*) når det drejer sig om vedligeholdelse af brugere, organisatoriske enheder og roller/rettigheder. I AD parlør kaldes disse entiteter også for *AD objekter*.

Figur 1 og 2 nedenfor illustrerer hvorledes GKs AD strukturelt er opbygget mht. vedligeholde af enheder og brugere.



Figur 1 – GKs overordnede organisation repræsenteret i AD



Figur 2 – GKs Børn og Skole område indeholdende brugerkonto for direktøren på området

Mange IT-fagsystemer benytter sig af et tilsvarende koncept for brugere, organisatoriske enheder (afdelinger), roller/rettigheder osv. for at kunne understøtte den opgave de er udviklet til. Oplysninger om disse entiteter oprettes og vedligeholdes typisk manuelt via en administrativ brugergrænseflade, som fagsystemet stiller til rådighed.

Det er ikke hensigtsmæssigt, at skulle genindtaste oplysninger om brugere, enheder, osv. i IT-fagsystemer, når de samme oplysninger allerede er oprettet (og vedligeholdes fortløbende) i AD. Udover at der tale om en ekstra manuel opgave, er der samtidig risiko for inkonsistens mellem AD og de (mange) systemer, som håndterer disse oplysninger.

Derfor er det en stående målsætning for GK, at IT-fagsystemer der opererer med brugere, enheder og roller/rettigheder genbruger de tilsvarende entiteter fra AD der allerede er oprettet dér, således at oplysninger er logisk identiske på tværs af AD og fagsystemer. Genbrug understøttes vel at mærke automatisk – uden manuel intervention. I den korte udgave; når vi én gang har oprettet en bruger, en enhed eller en rolle (gruppe), så skal vi ikke oprette entiteten igen i et fagsystem eller andre steder.

2.1 Integration med AD

Der findes flere standard teknologier for integration med AD.

Ift. GK er leverandører i udgangspunktet fri til at vælge den teknologi de foretrækker for integration af eget fagsystem til GKs AD. Det er dog en forudsætning af at den integration der udvikles, understøtter følgende overordnede krav:

1. Integrationen omfatter alle 3 typer af AD entiteter, dvs brugere, organisatoriske enheder (OU) og AD sikkerhedsgrupper
2. Ændringer til entiteter afspejles i realtid eller næsten realtid i fagsystemets kopi af de(n) tilsvarende entitet(er).
3. Alle operationer der er lovlige at udføre på entiteter i AD, afspejles "loyalt" i de tilsvarende entiteter i fagsystemet. Fx er der en forventning til, at objekter der slettes fra AD, også slettes fra fagsystemet, mv.
4. Integration kan ske uden såkaldte AD skema udvidelse.

Mere detaljeret tager brugeradministration af fagsystemer afsæt i følgende principper:

5. Brugere der allerede er oprettet på GKs IT-platform, skal ***ikke*** oprettes igen (ifm implementering af nyt IT-system), men afspejles automatisk ved passende integrations mekanisme i fagsystemet. Tilsvarende for organisatoriske enheder. Eventuelt for roller (AD sikkerhedsgrupper).
6. Nye oprettelser, ændringer til (herunder flytninger) og sletninger af brugere, organisatoriske enheder og roller (sikkerhedsgrupper), afspejles 100% i fagsystemer ***uden manuel interaktion***.
7. Nye oprettelser, ændringer til (herunder flytninger) og sletninger af brugere, organisatoriske enheder og roller (sikkerhedsgrupper), afspejles 100% i fagsystemer ***med det samme*** (nær realtid).
8. Rollebaseret autorisationsstyring administreres i udgangspunktet i AD (via AD sikkerhedsgrupper) – ***ikke*** i fagsystemet. Hvis fagsystemtets funktion er afhængigt af, at der kan administreres roller (og tildeling af roller til brugere) i fagsystemet, skal der etableres integration til AD, så fagsystemets administration *skrives igennem* til AD.
9. Rettighedstildeling til funktioner, data og andre IT-ressourcer tildeles altid til en given bruger via AD gruppe medlemskab. Mao – det skal være muligt alene ved konsultation af AD, at afgøre hvilke rettigheder der er tilknyttet en specifik bruger eller gruppe af brugere.
10. Administration af brugere, brugeres adgang til fagsystemer og brugernes rettigheder i fagsystemet skal kunne ske alene med udgangspunkt i AD (altså uden kendskab til fagsystemets administrative grænseflade).
11. Rettigheder kan tildeles via indirekte AD gruppe medlemskab (grupper i grupper i grupper i ...).
12. Løsninger der integrerer med AD understøtter SSO. Det vil sige at brugeren ikke skal logge eksplicit ind i fagsystemet, hvis brugeren allerede via oprettelse i AD og medlemskab af passende fagsystem

specifikke rettighedsgrupper, kan anvende fagsystemet uden yderligere tiltag (selvfølgelig først når brugeren er autentificeret ved login på Windows skrivebord – eller tilsvarende).

Gennem flere år med tilhørende IT anskaffelser, har GK oplevet en stor variation i funktion og kvalitet i de AD–fagsystem integrationer, som vi er blevet præsenteret for. Typisk er det et hjørne af IT-løsningen som leverandører ikke har meget fokus på – det er trods alt IT-systemets forretningsvendte funktion og hjælp til understøttelse af allehånde forretningsprocesser, der giver værdi.

Som konsekvens har GK forestået design og udvikling af en åben og fleksibel integrationsservice, som leverandører kan vælge at benytte, med henblik på at opfylde de AD–fagsystem integrationskrav der er beskrevet ovenfor.

Servicen kaldes for **ADEventService**.

Servicen udemærker sig ved:

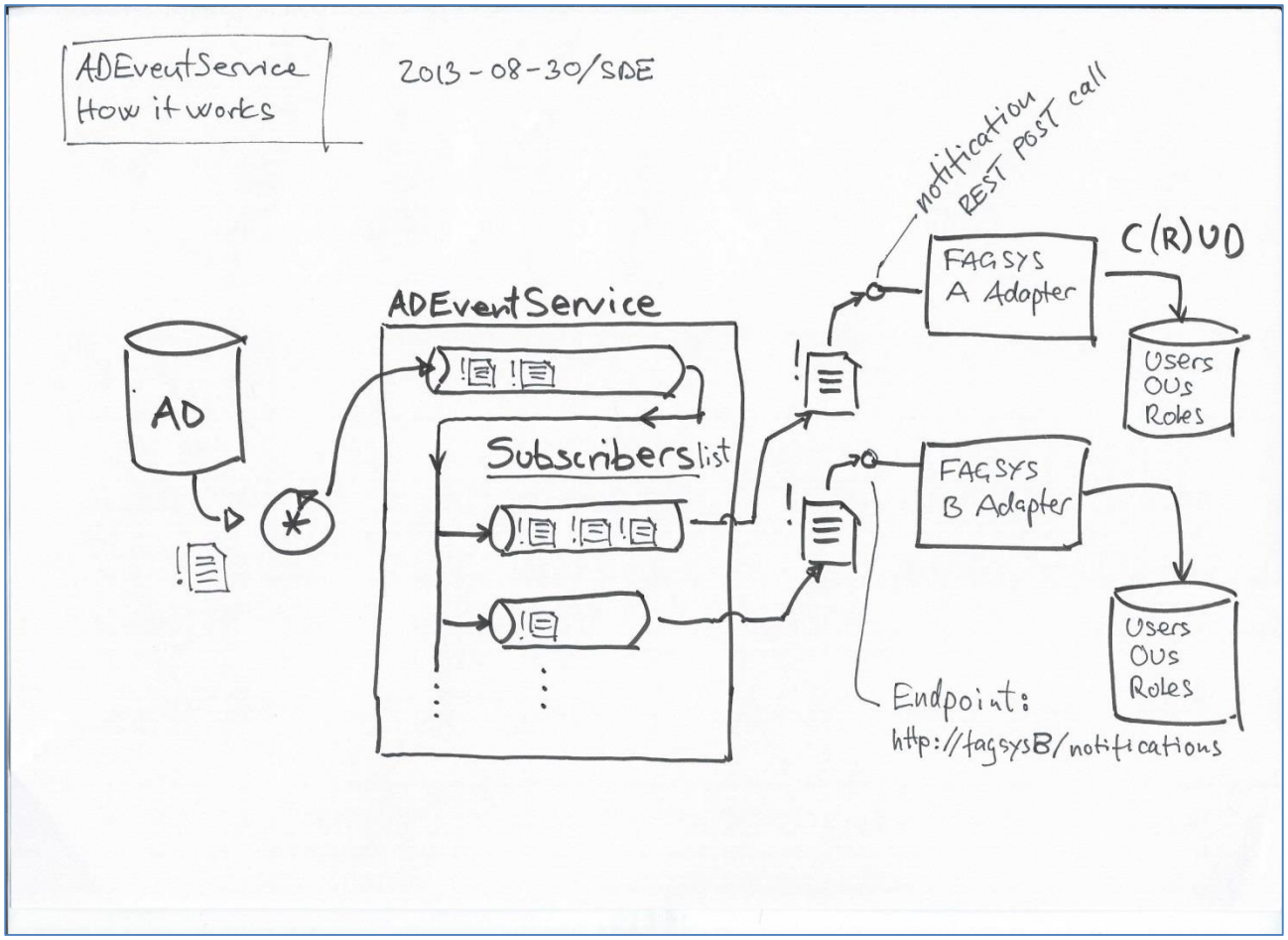
13. at etablere en klar adskillelse af det tekniske og aftalemæssige ansvar mellem GK og leverandøren af fagsystemet
14. at kunne implementeres i fagsystemer med en begrænset indsats
15. at basere sig på moderne web baserede snitflade mønstre
16. at være driftsmæssig robust

ADEventService stilles efter nærmere aftale til rådighed for andre myndigheder uden omkostning.

I det følgende beskrives i (semi nørdet) detalje, hvordan fagsystemer integreres med ADEventService.

3 How it works

ADEventService løsningen er illustreret i hosstående figur 3.



Figur 3, ADEventService AD hændelsesflow

ADEventServicen er implementeret som en Windows service, der (er) installeres(t) på en passende Windows (2008+) server. Servicen holder konstant øje med ændringer til 3 typer af objekter i AD, nemlig brugere (user), organisatoriske enheder (ou) og AD sikkerhedsgrupper (group).

Når en ændring registeres (fx oprettelse af en ny bruger) omdannes registreringen til en hændelse der samles op i en kø for senere behandling.

Et fagsystem der ønsker at få besked om ændringer til AD objekter (de 3 typer nævnt ovenfor), tegner et abonnement hos ADEventServicen. Når abonnementet tegnes oplyser fagsystemet ADEventServicen om en såkaldt adapteradresse som på vegne af fagsystemet, skal tage sig af de hændelser, der senere vil blive sendt til fagsystemet, dvs. adapteren. I figur 1, har fagsystem B fx angivet sin adapter adresse til http://fagsysb/notifications ved tegning af abonnementet.

Pt tegnes abonnementet ved at oplyse give GKs ITA-funktion besked om adressen på adapterendpointet.

Herefter og hvis abonnementet er aktivt, vil der for hver gang der sker ændringer til (de 3 typer af) objekter i AD, POST'es en besked til adapteradressen (adapter endpoint). Beskeden indeholder metadata om beskeden selv (besked ID, afsender, tidsstempel og typen af besked) og desuden data om det objekt i AD der er blevet ændret.

Det er nu adapterens opgave, at udrede/afgøre om beskeden og det objekt beskeden har med som nyttelast, overhovedet er interessant. Det er ikke ualmindeligt at modtage beskeder, som ikke repræsenterer reelle ændringer i AD objektets forretningsrettede data. Det hænger sammen med at mange ændringer til AD, er ret tekniske, som fx opdatering af tidsstempel for næste kodeordsskift osv. I de tilfælde, hvor der ikke er tale om reelle ændringer vil adapteren sikkert blot reagere med at droppe hændelsen.

Omvendt, når der er tale om en reel ændring (fx oprettelse af en ny bruger), er det adapterens ansvar at opdatere fagsystemets lokale repræsentation af brugeren i fagsystemets database.

Netto – når der POSTes til adapterens endpoint, *overdrages ansvaret* for hvad der skal ske med ændringen til adapteren/fagsystemet.

Konceptet er således ret enkelt – ADEventService finder ud af, at der er sket noget med et objekt i AD, giver besked til en fagsystemsadapter, der gør noget ved sagen (eller lader være).

Kompleksiteten opstår først, når ADEventServicen og mere generelt, den type af (EDA mønster baserede) integrationsmotorer, skal monteres i virkeligheden – en virkelighed der jo er udsat for nedbrud af netværksforbindelser, ustabile servere og andet skidt.

Derfor er ADEventServicen udviklet med anvendelse af en række robuste arkitektur mønstre. Servicen tager sig af, at levere beskeder til fagsystemer ***pålideligt***. Hvis ADEventServicen ***selv***, en eller flere fagsystem adapter(e), en netværksforbindelse osv. er nede, venter servicen blot til at skidtet virker igen og fortsætter derefter med at sende beskeder til abonnenter. Ændringer der er opstået mens servicen, adapter mv. har været utilgængeligt er altså ikke gået tabt – de ligger bare i kø til at blive behandlet, når mekanikken virker igen.

En fordel med ADEventServicen er at den kun sender besked om ændringer for entiteter, når de faktisk er (formodet) ændret – og at det sker med det samme. Nærmere bestemt, når ADEventService registerer en ændring i AD, går der kun få sekunder, før beskeden er leveret videre til fagsystemadapteren. Med andre ord, ADEventSevice kan medvirke til at ændringer i AD afspejles i fagsystemer i næsten realtid – og dermed medvirke til større transparens og mindre usikkerhed i de arbejdsgange, der fx vedrører administration af brugere, organisatoriske enheder og roller/grupper.

4 How to ... udvikling af adapter

Hvordan implementeres integration til ADEventServicen så konkret?

Grundlæggende består opgaven i, at implementere en adapter som beskrevet ovenfor. Adapteren stiller én REST baseret POST metode til rådighed, som kunne se ud som følger ...

```
namespace ADChangeDemoSubscriber
{
    // ============================================================================
    public class NotificationsController : ApiController
    {
        // ----------------------------------------------------------------------------
        // POST api/notifications
        public HttpResponseMessage Post([FromBody]string value)
        {
            try
            {
                ADEvent dtoADEvent = GK.AD.DTO.Serializer.DeserializeFromJson<ADEvent>(value);

                if (dtoADEvent.ADObject != null)
                {
                    IADObject adobj = dtoADEvent.ADObject;
                    if (adobj != null)
                    {
                        // Afgør om beskeden (dtoADEvent) og det objekt (adobj)
                        // som beskeden har med som nyttelast, overhovedet er interessant.
                        // GØR noget fornuftigt, hvis det er tilfældet !!!
                    }
                }

                return Request.CreateResponse(HttpStatusCode.OK);
            }
            catch
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, "Upps!");
            }
        }
    }
}
```

Post (POST) metoden i kodeeksemplet modtager en streng som repræsenterer en hændelse af type ADEvent. Strengen deserialiseres til et objekt af typen ADEvent (som implementerer følgende interface) ...

```
namespace GK.AD.DTO
{
    // ================================================================================
    public interface IADEvent
    {
        string CorrelationID { get; set; }
        WhoWhatWhen SenderInfo { get; }
        IADObject ADObject { get; set; }
        ADEventType ADEventType { get; set; }
    }
}
```

Hændelsesobjektet indeholder attributten ADObject (af type IADObject) som efter udpakning ovenfor indeholder et user, ou eller group (DTO) objekt.

I nogle tilfælde kan ADObject attributten være null, så der skal altid testes for om det er tilfældet, før attributten bruges.

Nedenfor er interfacet for user listet …

```
namespace GK.AD.DTO
{
    // ================================================================================
    public interface IADObject
    {
        ObjectClass objectClass { get; set; }
        string objectGuid { get; set; }
        string dn { get; set; }
        string canonicalName { get; set; }
        string name { get; set; }
        string description { get; set; }
        string displayName { get; set; }
        bool isDeleted { get; set; }
    }
}
```

```
namespace GK.AD.DTO
{
    // ================================================================================
    public interface IUser : IADObject
    {
        string sAMAccountName { get; set; }
        string userPrincipalName { get; set; }
        string givenName { get; set; }
        string initials { get; set; }
        string sn { get; set; }
        string title { get; set; }
        string manager { get; set; }
        string department { get; set; }
        string streetAddress { get; set; }
        string physicalDeliveryOfficeName { get; set; }
        string mail { get; set; }
        string telephoneNumber { get; set; }
        string mobile { get; set; }
        string objectSID { get; set; }
        bool AccountLockedOut { get; set; }
        bool AccountEnabled { get; set; }
        bool PasswordExpired { get; set; }
        bool DontExpirePasswordEnabled { get; set; }
        DateTime AccountExpiresDT { get; set; }
        string extID { get; set; }
        string shortKey { get; set; }
    }
}
```

Når adapteren modtager en hændelse (ADEvent) inklusive et AD objekt (IADObject) som nyttelast, skal der ske følgende:

1. Undersøg om det er en hændelse og et objekt der overhovedet skal gøres noget ved?
2. Hvis i givet fald, gør noget – typisk opdatér fagsystemets lokale database med det tilsvarende objekt
3. Returnér OK eller fejl som HTTP response fra POST kaldet

Hvordan adapteren afgør om en ændring af data for et AD objekt er interessant for fagsystemet eller ej, kan tage mange former.

Én praksis i GK er at forespørge AD om medlemskab af de sikkerhedsgrupper (roller) som er oprettet i tilknytning til installation af fagsystemet.

Eksempel:

IT-fagsystemet FagsysABC har defineret følgende roller (dvs oprettet følgende AD sikkerhedsgrupper):

FagsysABC-admin, indeholder brugere der er administratorer af FagsysABC.
FagsysABC-default-user, indeholder sagsbehandlere der anvender FagsysABC
FagsysABC-guest, indeholder brugere med kun læseadgang til FagsysABC

De 3 grupper er ***medlem af*** en sidste 4. gruppe ...

FagsysABC-member

For at afgøre om en brugerændring er relevant for fagsystemet FagsysABC , kan adapteren nu forespørge AD om brugeren er medlem af sikkerhedsgruppen FagsysABC-member. Denne forespørgsel kan udføres ved at benytte de API'er som AD stiller til rådighed (fx UserPrincipal.GetAuthorizationGroups() i .NET) eller anvende de de (WCF baserede) web services som GK stiller til rådighed (IsUserInRole(string userID, string role)).

Repræsenterer det brugerobjekt der medbringes i hændelsen til adapteren, en bruger der er medlem FagsysABC-member og dermed indirekte medlem af en af de 3 grupper ovenfor, så oprettes eller rettes brugeren i det her tilfælde i FagsysABCs lokale bruger database.

I nogle tilfælde, kan hændelsen også indeholde et objekt der er blevet slettet (isDeleted == true) og hændelsen selv er også stemplet med hændelsestypen slettet (ADEventType == Delete).

Igen, hvis det er et objekt det er interessant at gøre noget ved – i den her situation afklares det sikkert ved at undersøge om objektet allerede er oprettet i fagsystemets lokale database – ja, så slettes objektet fra fagsystemets database – eller markeres som slettet.

Det er altså adapterens opgave, for objekter der er interessante, at oprette, ændre/rette og nedlægge de tilsvarende kopiobjekter i fagsystemets database, konfiguration eller hvor den slags data nu opbevares.

Når adapteren opretter, retter og sletter objekter (brugere, organisatoriske enheder og evt grupper) i fagsystemets database er det vigtigt at kopiobjektet i fagsystemdatabasen, oprettes med en reference/fremmednøgle til det tilsvarende objekt i AD. Nøglen skal bruges til at genfinde objektet i den lokale database ved efterfølgende opdateringer til objektet.

Her anbefales det på det kraftigste at benytte feltet (AD attributten) objectGuid som nøgle. objectGuid attributten er garanteret at være unik i objektets levetid og er samtidig defineret for alle AD objekter.

Sagen er, at andre felter som kunne ligne nøgler – ikke er det pr. AD definition. Fx er en brugers loginnavn (sAMAccountName) sikkert fristende at bruge som nøgle i mange fagsystemer (fordi nøglen i forvejen bliver brugt internt af fagsystemet) ***men det skal man undgå***. Årsagen er, at selv om det frarådes af MS, så er det faktisk lovligt at omdøbe loginnavn. Da alle AD operationer skal spejles 100% i fagsystemet (for at sikre konsistens) skal omdøbning af loginnavn derfor accepteres af fagsystemet.

4.1 Fejlhåndtering

Det er god praksis at returnere en HttpResponseMessage fra POST metoden. Hvis beskeden accepteres af adapteren – og det gælder også i det tilfælde, hvor adapteren bare dropper beskeden, returneres HTTP status kode OK eller Accepted (svarende til 200/202). ADEventService gensender ikke beskeden.

Hvis derimod beskeden ikke kan behandles, dvs der opstår en ikke håndteret fejl, returneres en BadRequest fejlkode. ***ADEventService gensender IKKE beskeden***, men logger fejlen – evt med den meddelse, der returneres fra adapteren (i eksemplet ”Upps!”) – til hjælp for fejlfinding.

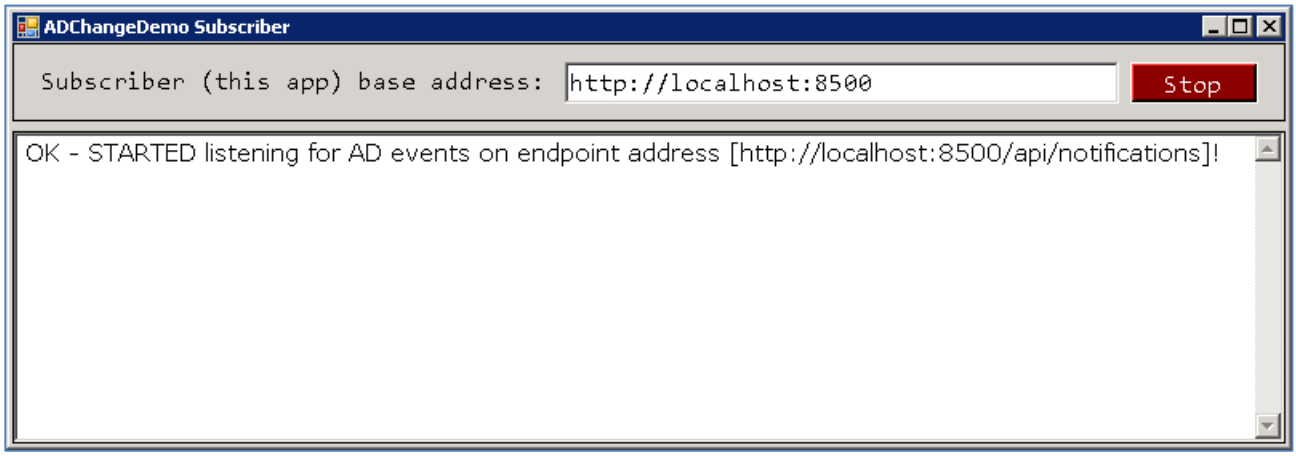
Når en adapter ikke er aktiv/tilgængelig, vil ADEventServicen forsøge at sende og gensende beskeder med passende mellemrum. Når adapteren vågner op til dåd igen, sendes i kronologisk rækkefølge alle de beskeder der har hobet sig op, siden adapteren gik offline. Har en adapter været inaktiv i længere tid, kan der gå længere tid, før ADEventServicen begynde at levere beskeder igen, selv om adapteren er blevet aktiv.

En adapter har mulighed for sende besked til ADEventService for at give servicen besked om, at adapteren er aktiv igen – i stedet for at skulle vente på, at ADEventService selv finder ud af, at adapteren er klar igen. Dette scenarie (og andre adapter-ADEventService kontrol operationer) beskrives ikke yderligere.

---

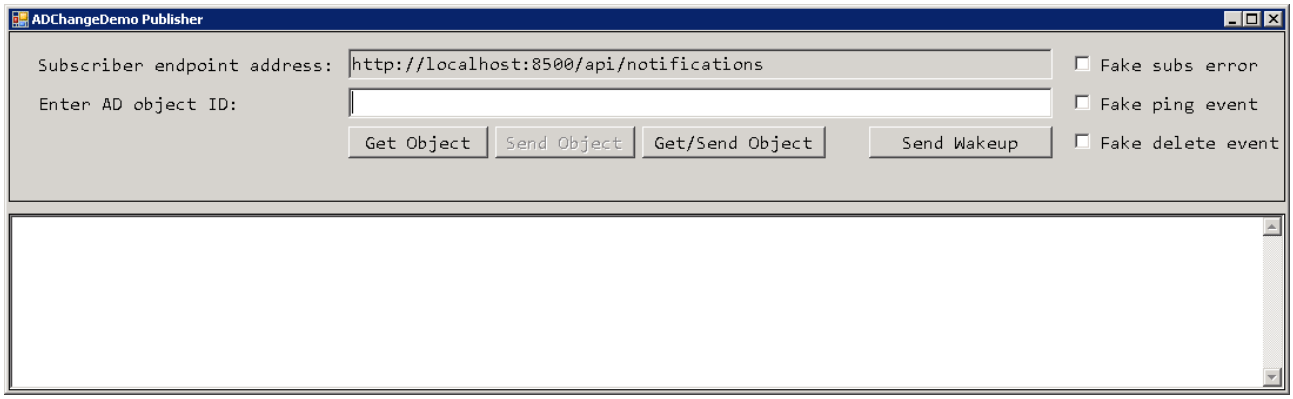
5 How to ... udvikling af adapter (2)

For at lette implementering af adapter, har GK udviklet et adapter demo applikation kaldet ADChangeDemoSubscriber...



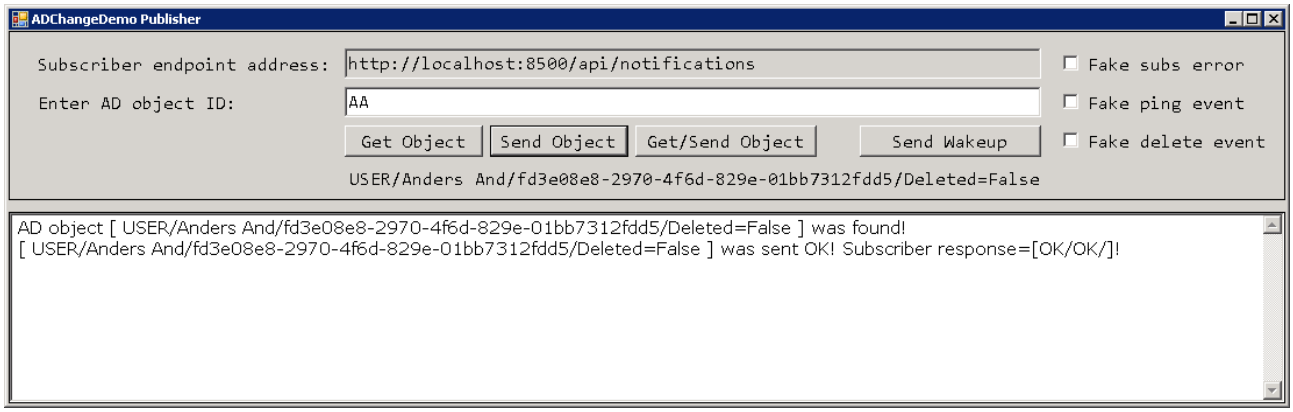
Demo applikationen leveres med fuld kildekode som et Visual Studio 2012 projekt (solution). Leverandøren er velkommen til at tage udgangspunkt i demoen inklusive kode, ved bygning af adapter til eget fagsystem.

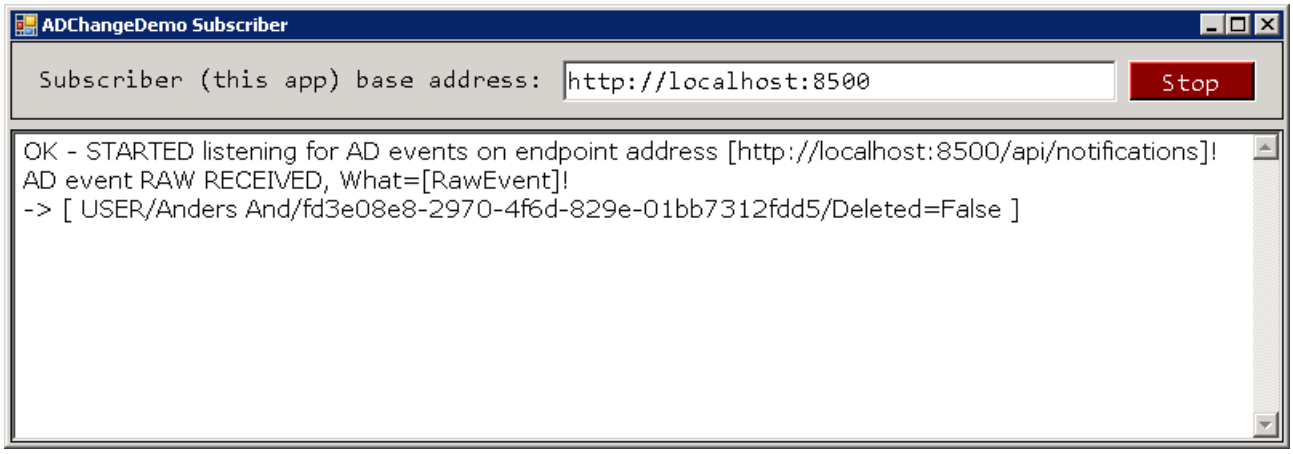
For at teste kald fra ADEventService er der endvidere udviklet en anden demo applikation – ADChangeDemoPublisher – der simulerer ADEventServices funktion...



ADChangeDemoPublisher applikationen afvikles på en maskine, der er meldt ind i et AD domæne. Indtast et søgekriterie i "Enter ... ID" feltet (det kan fx være et AD brugerlogin) og klik derefter på "Get Object" knappen.

Når (hvis) brugeren er fremfundet, kan en opdatering af objektet og forsendelse til adapteren simuleres ved at klikke på "Send Objekt" knappen. Nedenfor er vist fremsøgning af AA med efterfølgende simulering ...





5.1 Initielt load af objekter til fagsystem

Inden et nyt IT-fagsystem sættes i drift, skal det normalt tankes op med de brugere, enheder, roller osv, som systemet skal anvende.

Hvordan sker det i et hændelsebaseret koncept, som ADEventServicen?

Løsningen er ganske enkelt, at ”skubbe” til de objekter, som skal oprettes i fagsystemet. Hvis der skal oprettes enheder, skubbes der først til dem, med øverste enhed i hierarkiet først, dernæst enheder på næste hierarki niveau, dernæst tredje osv.

Efterfølgende skubbes til de brugere der skal monteres ind i løsningen.

Hvis der er tale om få objekter, kan passende AD administrationsværktøjer bruges interaktivt til at skubbe til enheder, brugerne osv. men uden at ændre noget.

Er der tale om mange brugere/enheder osv. kan skubberiet udføres med passende script mod AD.
