1.6 Type integrasjoner

Ulike typer integrasjoner med tilhørende fordeler, svakheter og muligheter.

Ideelt sett skulle alle integrasjoner du skal utvikle, implementere eller oppgradere vært basert på moderne arkitektur med tilhørende fleksible API, men rammebetingelsene er ofte ikke slik. Eksisterende system er planlagt å leve videre i flere år og API-ene som evnt finnes er mangelfulle, statiske, dårlig dokumentert, full av feil (bugs), lite tilgjengelig kompetanse hos system-leverandøren, osv. Konsekvensen av dette kan bli at en integrasjon enten er basert på forskjellige typer teknologier eller av praktiske årsaker løsninger som er basert på gammel teknologi. Prosjektet og beslutnings-takere må derfor ha kunnskap om flere typer teknologier, med tilhørende innsikt i styrke, svakheter, muligheter og trusler innen alternative løsninger.

Vi vil her ser på følgende type løsninger:

- Database-kobling
- Manuell Import & Eksport
- Automatisert Import & Eksport
- Periodiske rapporter
- Applikasjonsbasert integrasjon
- APP/WEB-integrasjon
- API til API integrasjon
- Abonnement

Synkron

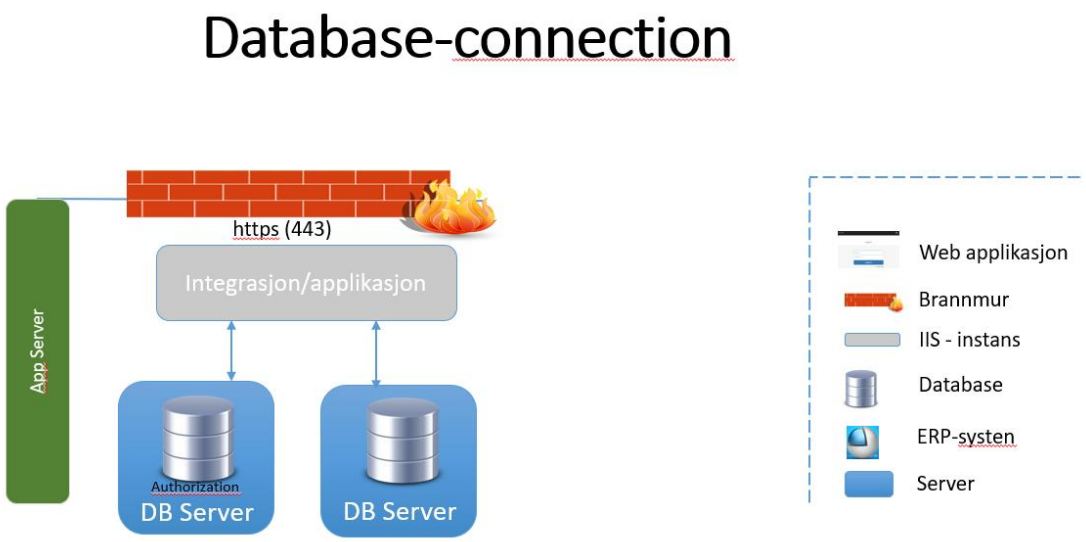
Noen av løsningen er basert på sanntids-integrasjoner (realtime), hvor brukeren gjerne benytter grensesnitt (skjermbilder) som er koblet mot flere applikasjoner/databaser. Vi kaller dette synkrone løsninger.

A-synkron

Andre system forutsetter at du kun kan lage løsninger mot API eller databasen, hvor informasjonen du skal overføre kun er basert på meldinger/filer du selv ikke kan påvirke, noe som medfører asynkrone løsninger.

Database-kobling

Da SQL-baser (standard query language) ble utbredt, bla pga lavere priser med Sybase/MS-SQL vs tyngre plattformer som DB2 og Oracle, kom det teknologi for enkel tilkobling til databasen. Verktøy som ODBC, OleLink, etc gjorde det enkelt å lage tilkoblinger fra egenutviklede applikasjoner og regneark. (MS Access, DataEase, Visual Basic, Lotus og Excel). En kan da rimelig effektiv ha direkte tilganger til tabeller i SQL-basen for å vise eller oppdatere innholdet i databasen. Vær oppmerksom på at oppdateringer rett mot SQL-basen er av stor risiko for at forretningslogikken i applikasjonen eller databasen ikke blir ivaretatt.



Koblinger mot databaser kan gjerne kombineres med programmer lagret i SQL i form av det vi kaller prosedyrer, views og triggere. En stor utfordring med å migrere gode løsninger fra OnPrem (lokal database) til Cloud (sky), er at denne type database-logikk ikke er tilgjengelig og må erstattes med andre løsninger. Faktisk er dette så stort ett problem at OnPrem-løsninger som Visma Business og Global med Windows-baserte grensesnitt, vil kunne leve i mange 10-år.

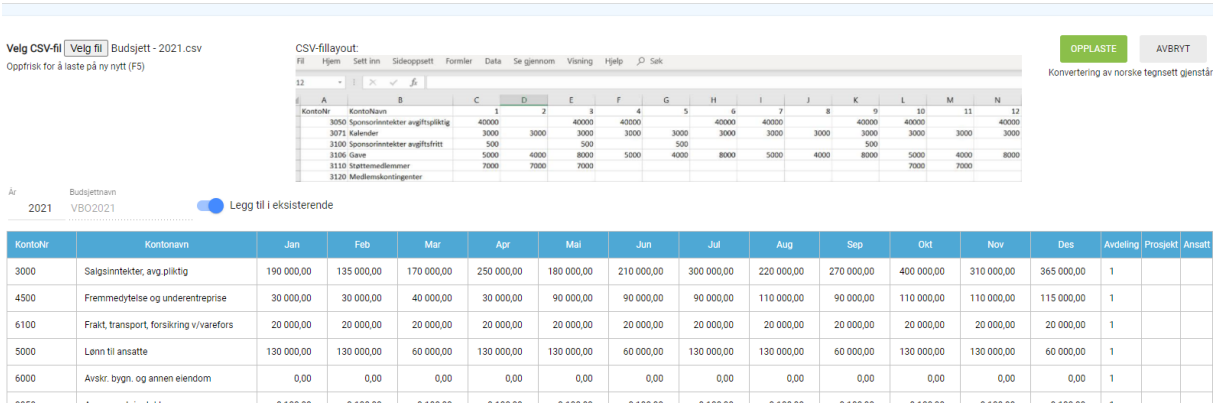
Tilpasninger av applikasjon og database-connection-integrasjoner er også en årsak til at sky-migrering bli kostbart, utsatt eller uaktuelt nærmeste 5 år. Opprettholdelse av kompetansen på denne typen integrasjoner og tilpasninger blir da svært sentralt, noe som gjelder både hos driftsansvarlig og hos systemleverandør.

Manuell Import & Eksport

I noen tilfeller ønsker en ikke helautomatiserte integrasjoner, da overførte data gjerne skal kontrolleres eller brekkes om, før de importeres eller eksporteres. Ombrekkingen kan være kode-verdier fra eksternt system som skal lagres med andre koder eller format i mottakende system (mapping). Manuell løsningen utvides med automatikk når behovet for kvalitetssjekk ikke er nødvendig, hvor mulige feilkilder og behov for manuelle korreksjoner er fjernet.

Ett eksempel på import, er opplasting av budsjett fra excel-csv-fil eller ordre-transaksjoner til Økonomi/ERP-system. Import/Excel-fila kan da være sendt fra eksterne program eller laget manuelt i Excel, hvor layout og innhold i fila kan variere fra gang til gang. En annen årsak kan være at grunndata mangler, for eksempel konto, avdeling, prosjekt eller ansatt i mottakende system, noe som må opprettes før importen kan fullføres.

I slike tilfeller kan en produsere en logg-fil eller lage en brukervennlig applikasjon hvor en velger input-fil, sjekker mot grunndata og visualisere dette i skjermbilde, før en velger å importere/oppdatere.



KontoNr	Kontonavn	Jan	Feb	Mar	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Des	Avdeling	Prosjekt	Ansatt
3000	Salgsinntekter, avg.pliktig	190 000,00	135 000,00	170 000,00	250 000,00	180 000,00	210 000,00	300 000,00	220 000,00	270 000,00	400 000,00	310 000,00	365 000,00	1		
4500	Fremmedytelse og underentreprise	30 000,00	30 000,00	40 000,00	30 000,00	90 000,00	90 000,00	90 000,00	110 000,00	90 000,00	110 000,00	110 000,00	115 000,00	1		
6100	Frakt, transport, forsikring v/varefors	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	20 000,00	1		
5000	Lønn til ansatte	130 000,00	130 000,00	60 000,00	130 000,00	130 000,00	60 000,00	130 000,00	130 000,00	60 000,00	130 000,00	130 000,00	60 000,00	1		
6000	Avskr. bygn. og annen eiendom	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1		

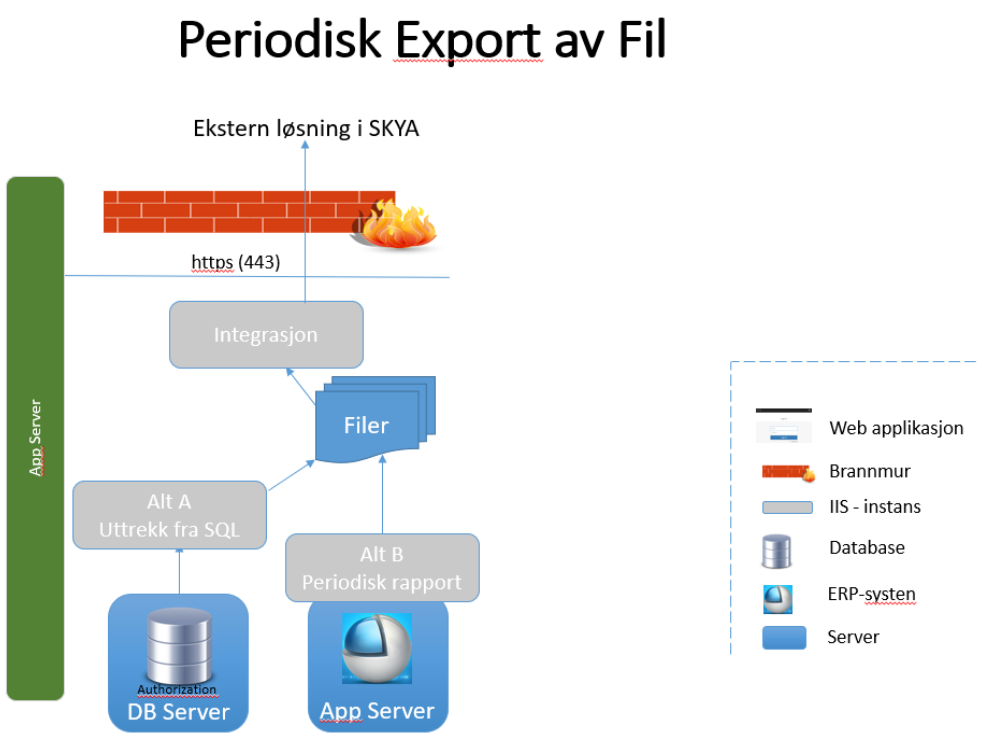
Eksempel på Budsjett-import, hvor en kan styre overskrivning eller føye til eksisterende budsjett. Behandles under i kap 3.5 *Import-funksjon mot API*.

Tilsvarende kan en ha på Eksport, gjerne med valg av fil-format og hvilke felt/tabeller som skal benyttes.

Automatisert Import & Eksport
Når denne type import/eksport er godt uttestet og det normalt aldri er behov for feilhåndtering, kan løsningen lages som tidsstyrte prosesser. Noen standard-system har ferdige løsninger for dette i form av «edi-klokke» eller «periodiske rapporter». Det betyr at prosessen startes jevnlig, for eksempel hver time, eller ved angitte tidspunkt.

Hvis en har laget en egen applikasjon, kan lesing, behandling/mapping og skrivning til mottakende system skilles ut som egen funksjon/modul, uten WEB/APP-grensesnitt. Programmet en da sitter igjen med kan så startes periodisk med en integrasjonsportal, Windows Scheduler eller trigges med en funksjonsknapp i en applikasjon.

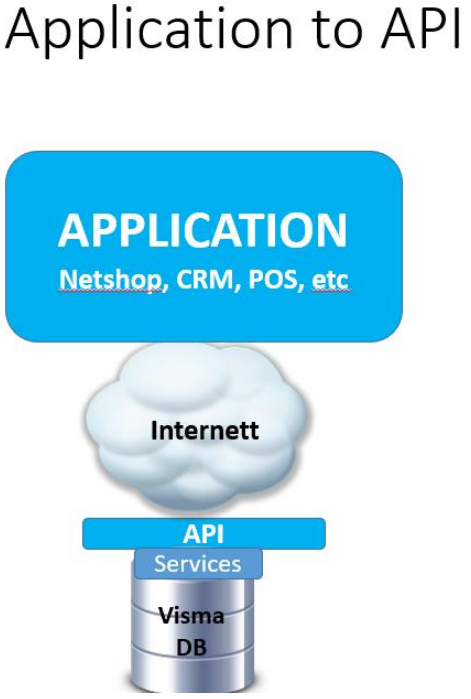
Hva kan være årsaken til at en må lage eksporter og integrasjoner via filer, selv når systemet har rimelig komplette API?
Ett eksempel er fra ett kundesystem (KIS) hos ett Kraftselskap. Her skal en hente ut kunder som har blitt endret siste døgn, noe som hadde vært enkelt om en bare trengte å sjekke mot siste endret tidspunkt i kundetabellen (lastChangeDateTime) eller benytte «webhooks» hvis så fantes.
Utfordringen er at kunderelaterte endringer hos et Kraftselskap skjer flere plasser i databasen. Det medfører at ikke bare stamdata på kunde må overføres, men også produkter (tariffer), strømforbruk, nye målere, bytte av nett-leverandør, mm. Et uttrekk fra systemet sitt standard API løser ikke saken med mindre systemleverandøren kunne tilby en egen service til formålet. Løsningen ble da å benytte enten eksport-funksjoner i form av periodiske rapporter som kan tidsstyres eller i verste fall lagre egne program som går direkte mot SQL-databasen.



Se case Hansen EnoroCX vs APSIS i kap x.x

Applikasjonsbasert integrasjon (synkron)

En effektiv og driftssikker løsning oppnås hvis integrasjonen kan implementeres direkte i applikasjonen. F.eks Netthandel som skal kobles til et verts-system som kan vært et ERP-system. Netthandel-rammeverk for denne type løsninger kan være Wordpress eller WooCommerce, mens standardsystem for kundesystem (CRM) kan være Salesforce eller MicroSoft Dynamics. Disse systemene har fleksible bibliotek/funksjonalitet for å utvikle denne typen integrasjoner mellom egen løsning og eksterne API.



Alternativt kan en skreddersy integrasjonen i egen applikasjon hvis en sitter på kildekoden eller moduler for dette. Det gir stor frihet i utforming av logikk, for eksempel hva skal være styrende ved opprettelse (kundenr), direkte-oppsalg og synkronisert innhold. Dette kan gi realtime visning av kunde/lager-saldo, pris pris-kalkulasjoner utført av ERP-systemet. Skreddersydde integrasjoner inne i selve applikasjonen åpner for langt bedre muligheter for sikker og brukervennlig feilhåndtering.

Med en applikasjon programmert direkte mot ett API vil en kunne behandle/presentere resultat av aktiviteter (request), utført mot eksternt API. Vi kaller gjerne dette for synkron integrasjon. For eksempel vil en vellykket opprettelse av ordre, returnere et ordrenummer fra ERP-systemet, som så kan lagres som referanse i overførte ordre i Netthandel. Hvis opprettelsen av ordren feiler, grunndata som ikke matcher (produkt, valuta, landkode, etc), vil responsen fra API-requesten kunne presenteres for sluttbruker, som umiddelbart kan korrigere innholdet og sende på nytt. Å løse tilsvarende med meldingsbasert (asynkron) er også i teorien mulig, men svært krevende, hvor en liten tidsforsinkelse kan medføre at brukeren har lukket skjermbilde før feilmeldingen vises.

Finnes det allerede en melding/edi-basert løsning som nesten aldri feiler, uten krav om direkteoppslag og som dekker funksjonelle behov i en integrasjon, kan den fint beholdes med mindre virksomheten av ikt-strategiske årsaker vil over på API-baserte integrasjoner.

APP/WEB-integrasjon
En annen variant for å sy sammen applikasjoner på, spesielt WEB-applikasjoner er det vi kan kalle en URL-integrasjon. I den mest primitive løsningen legger en inn en URL-link i ett menyvalg, for eksempel https://www.altinn.no/ og brukeren slipper å huske/lagre en kompleks link.

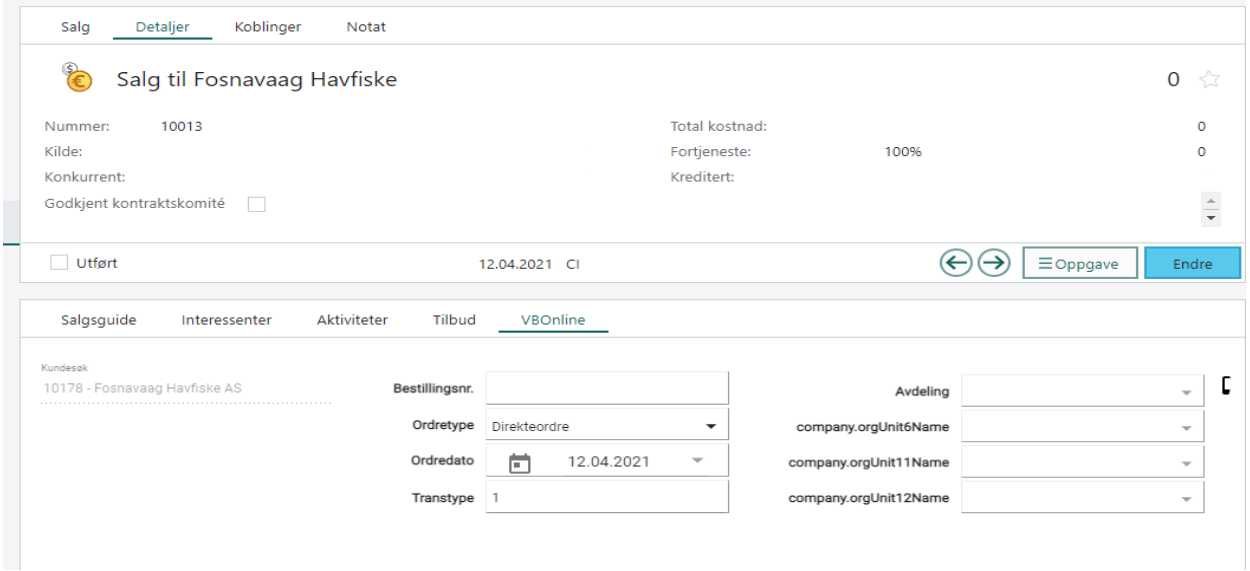
URL-integrasjoner kan så foredles med det vi kaller singel sign-on, context-håndtering eller tilpassede grensesnitt.

Context via url
En tar med parameterverdier fra vertssystemet som brukernavn, kundenr, saksnr, referanse, etc med inn som en del av url-strengen, som igjen tolkes av url-integrert applikasjon. Selve URL-strengen kan skjules for brukeren, men den vil kunne «sniffes» av en hacker og en må i en slik løsning ikke sende med sensitive data.

Context via local storage
En kan også se for seg at parameterverdier, for eksempel bildefiler (pdf) som ikke kan sendes via URL, blir lagret i «local storage» på PC/Mobil, for deretter å bli plukket opp av integrert system. Typisk kan være at en skanner bilder/strekkode med en Native Mobil-app, starter en WEB-applikasjon og henter inn bilde/strekkode fra lokal storage.

Single sign-on
En sømløs URL-kobling får en ved å unngå inn-logging på integrert applikasjon, pålogging skjer kun i vertssystem eller protal. En slik løsning forutsetter bruk av teknologi for krav til lagring av brukertilganger, for eksempel MicroSoft sin Active Directory. Behovet for dette er redusert hvis brukernavn sendes med i linken og nettleser kan lagre sist brukte passord. Da er det kanskje kun nødvendig å logge seg på integrert system en gang pr dag. Hvis pålogging skjer med bruk av det vi kaller «token», dvs en nøkkel-kode som kan gjenbrukes for en periode, kan det også medføre at pålogging kun skjer når nøkkel sin gyldighet er utløpt.

Grensesnitt
Eksempelet under viser hvordan CRM-systemet SuperOffice er URL-integrert med ett ERP-system, hvor bla kunde tas med inn i Opprett Ordre-bilde. Grensesnittene (skjermbildene) er laget så likt at brukeren knapt vet at det her jobbes i 2 forskjellig system.

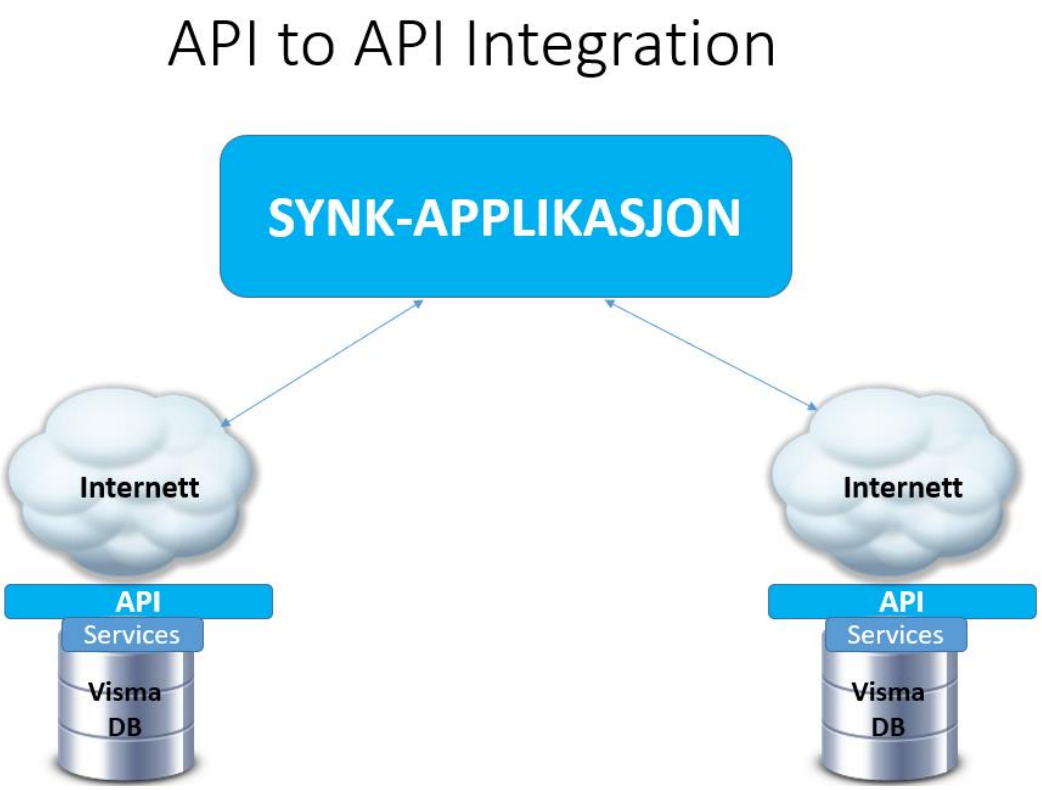


API til API basert integrasjon

Systemleverandørene kan begge tilby API. Det kan sammenlignes med at du har fergekai i begge ender, men må lage ferga som skal håndtere transportere mellom endepunktene.

Utgangspunktet kan vært at vi skal integrere system som ikke har rammeverk for integrasjon eller muligheter for tilpasninger/utvikling, hvor en kun har 2 standard-api tilgjengelig. Eksempel kan være oppslag i Brønnøysundregisteret som det ene systemet og et standard API for et skybasert ERP-system som mottakende system. Her kan en ikke alltid forvente at noen av aktørene tilbyr utvikling eller tilpasning, men kun integrasjonsgrensesnitt i form av standard-API. Det kan også være at løsningen skal være en mange til mange HUB for flere applikasjoner (trafikknutepunkt). API i et standardsystem kan kanskje utvides med nye servicer eller funksjoner på forespørsel, med svært variabel grad av fleksibilitet, kostnader og leveringstid hos aktuell systemleverandør. En risikerer at løsningen da er begrenset kun til de tjenestene som API-et kan tilby. Du kan for eksempel hente kundeinformasjon fra Brønnøysund, men ikke kundens epost fra API-et, selv om informasjonen skulle finnes i databasen.

Integrasjon mellom disse 2 API-ene må da løses med et mellomvare-program, enten synkront eller asynkront. Med asynkron menes at henting av data av ulike årsaker må mellomlagres før dataene lagres i mottakende system. Litt avhengig av hvor synkroniseringen kjøres, for eksempel hver 5.sek, kan dette oppleves tilnærmet synkron, men da også med fare for at respons kommer for sent (skjermbilde er lukket). Synk-applikasjonen kan i prinsippet ligge installert hvor en vil da den forholder seg til API som nås via internett. Videre må det settes opp en tidsstyrt prosess som angir når den skal starte og hvor ofte den skal repeteres, hvor for eksempel Windows Scheduler eller integrasjonsportaler kan benyttes.



ØB Portal er eksempel på Synk-portal

Det finnes portal-løsninger for denne type API-API-integrasjoner som har følgende funksjoner:

1) Automatisk dataoverføring mellom løsninger

Dette kan settes opp som en online løsning hvor dataoverføring skjer umiddelbart eller med tidsstyring, avhengig blant annet av hva slags løsninger som skal integreres og teknologi.

2) Konvertering av dataformater

I en del tilfeller må dataformater endres eller legges til for å bli håndtert riktig ved overføring. Integrasjonslogikk. Hvilke data skal overføres og hva som utløser dette. Portalen styrer hvilket system som "eier" grunndata, som betyr hvor de oppstår eller hvilket system som tildeler for eksempel kundenummer.

3) Tidsstyring av datautveksling

Vurderes i forhold til behov, datamengde og belastning på servere.

4) Logging av aktivitet

Ved feilsituasjoner kan detaljert logging være nyttig

5) Epost varsling

Det kan settes opp automatisk epost varsling ved spesielle hendelser.

6) Feilhåndtering

Det settes opp feilhåndtering som bestemmer hva som skjer hvis det oppstår feil – f.eks skal hele overføringen avvises, eventuelt kun transaksjoner med feil, automatisk retting av verdier, varsling av bruker mm.

Abonnere

Hente data fra en database, for eksempel kunder som er endret/nye/slettet siden sist synkronisering, kan være ressurskrevende om en snakker om store tabeller/baser og hentingen skal skje hyppig. Det betyr at hentingen tar tid og kan oppleves som asynkron, samt at vertssystemet blir unødig belastet. Moderne API-grensesnitt kan derfor tilby teknologier som betegnes som WEB-Hooks, Subscription og Publish. Her finnes det løsninger hvor API, som normalt er en passiv tjeneste, kan settes opp som aktive, hvor endringer sendes fra verts-system direkte til mottakende system sitt API og oppdateringer skjer realtime.

Se case VIC.AI mot ERP i kap 4.3 *VIC.AI – Portal API-API*