RIA WhitePaper

September 2018
Statens It

Hermed best practices for udvikling i Robotic Process Automation. Input er samlet fra Statens It, Statens Administration, Københavns Kommune samt Deloitte Danmark.

Indhold

Introduktion 2
Arkitektur 3
Fysisk infrastruktur 3
Virtuel infrastruktur (RIA-platform + VDI-klienter) .. 3
Robotkonti 4
Services 4
Attended (Understøttes ikke) – Front Office Robot 4
Unattended – Back Office Robot 4
Miljøer 5
Produktion 6
Prioritering 6
Roller 7
UiPath 7
Afvikling 13
UiPath 13
J-konto 13
Platform 15
UiPath 15
Udvikling 16
Introduktion 16
Standarder 16
Ansvar 16
Credentials 16
Deployment 17
Deployment af procespakke til robot 17
Best practice på RPA-platformen 18
Udvikling 18
Robotskelet – brug evt. state machines 22
Bilag 1 - Begreber 24

Introduktion

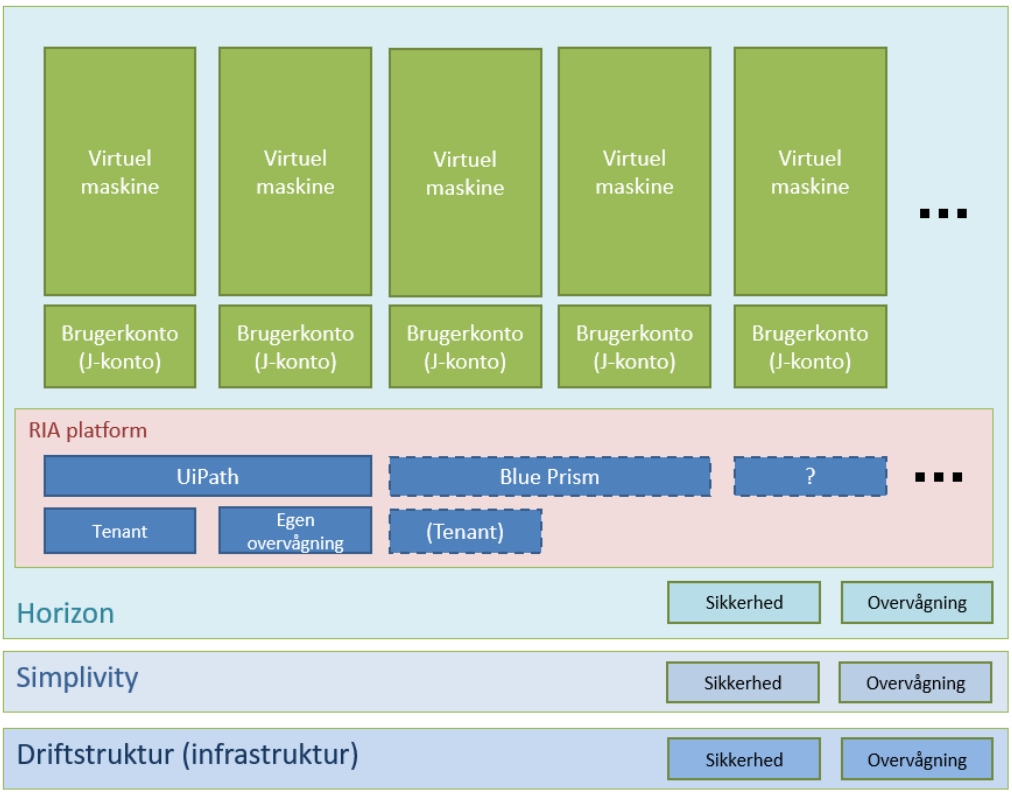
Dette dokument er en beskrivelse af Robotplatformen i Statens It - Robot It Arbejdsplads, forkortet RIA.

Vi forsøger at gennemgå de vigtigste emner på en kort og præcis måde så dokumentet kan tjene såvel som introduktion som en opslagsbog for processer og strukturer.

Dette er første iteration af dokumentet, så der er plads til forbedring. Kontakt evt. RIA-teamet hvis noget er uforståeligt eller mangler.

Arkitektur

Statens Its RIA-platform består af følgende komponenter:



Fysisk infrastruktur
I bunden af stakken ligger VMWare. Ovenpå denne ligger Simplivity og Horizon. Sikkerhed og Overvågning varetages af Statens It.

Kundens netværksmiljø (VLAN, FW, GPO og andre adgange) er kopieret til RIA-miljøet således at VDI'erne kan "se" præcis det samme som kundens SIA-pc'er.

Virtuel infrastruktur (RIA-platform + VDI-klienter)
Styringsserveren er en virtuel server der styrer sine klienter. Lige nu understøttes UiPath og Blue Prism er på vej. Hver kunde er administration (tenant) i sit eget hjørne af RPA-hotellet.

De **virtuelle klienter** er baseret på et image der minder så meget om en SIA-pc som muligt. Der er følgende undtagelser:

- GPO: der er ikke pauseskærm
- Windows: Windows Updates sker manuelt i servicevinduet

Lige som SIA-pc'er, så skal der bestilles åbning i FW hvis VDI-klienten skal tilgå særlige systemer, fx Navision. Ellers bør robotklienten kunne præcis det samme som kundens medarbejdere på netværket.

Bemærk, at *adgangen til systemer* er knyttet til den *robotbrugerkonto* (J-konto) som robotklienten afvikles med – og er altså ikke maskinafhængigt.

Robotkonti
Robotklienten afvikles med en særlig robotkonto: j-konto. Denne konto kan som udgangspunkt intet andet end at logge på en PC og er medlem af Domain Users.

På grund af sikkerhed er der ikke automatisk *Single Sign On* eller automatisk identitetsgenkendelse i systemerne. Skal en j-konto have adgang til et system, bestilles dette særskilt (mere herom i governancedokumentet).

Services

Attended (Understøttes ikke) – Front Office Robot
'Attended' automatisering afhænger af samarbejde mellem medarbejder og robot, hvor der kræves menneskelig input. Løsningerne er installeret på medarbejderens arbejdsstation og udløses af specifikke begivenheder, handlinger eller kommandoer. Fordi overvågning af automatisering ofte involverer medarbejdere, der bevæger sig mellem flere grænseflader eller skærme i en given transaktion eller kontekst, skal denne form for automatisering være fleksible og brugervenlige, således at medarbejderne fortsat kan arbejde mens processen eksekveres.

Understøttes ikke. Men vi kan ikke forhindre kunder i at have det kørende på deres egne pc'ere (evt. med egne licenser).

Unattended – Back Office Robot
'Unattended' automatisering afhænger af, at der ikke er menneskelig indgriben, og hvis der er indgriben, forgår det med kommandoer givet igennem Orchestrator. Denne form for automatisering er den absolut mest brugte form for automatisering. Det forventes af disse typer automatisering udelukkende styres gennem Orchestra-tor og der lægges op til at de eksekveres så effektivt så muligt, så licenser benyttes 24/7/365.

Miljøer

Det er muligt at det understøttes forskelligt i de underliggende RPA-teknologier, men som udgangspunkt understøtter RIA-platformen både DEV/TEST- og PROD-miljøer. Dette gøres primært ved at tildele den enkelte VDI-klient en rolle eller licens som angiver hvilket miljø den tilhører.

- UDVIKLING – stedet hvor man udvikler processen
  Adgang hertil har samtlige ressourcer for DEV-team.
- TEST – stedet hvor man tester processen med mere relevante omgivelse og data
  Adgang hertil gives til samtlige ressourcer for DEPLOY- og gerne DEV-team.
- PROD – den endelige deployment
  Overvågning af robotter via Orchestrator-team, evt. i samarbejde med SIT
  Overvågning og kunde

I de tilfælde kunden ikke har et testmiljø, så se afsnit "*Overvejelser ved udvikling i PROD"* på side 20.

Udvikling / Test

Som udgangspunkt leveres to RIA VDI'er (1 TEST og 1 PROD) per tenant. Det er tanken, at kunden udvikler og tester på den ene og sætter i PROD på den anden robotklient (RIA VDI). På denne måde er der altid en robot til udvikling og en ledig til test.

UiPath
I UiPAth angives DEV og TEST via de licenstyper som ligger i Administrationsmodulet i kundens tenant. Som udgangspunkt leveres både en TEST og en PROD licens per kunde. Der vil være en tilsvarende TEST- og PROD-VDI tilgængeligt.

DEV er gratis og varetages af Studio og er typisk en SIA-pc uden for RIA-platformen. Dog skal den registreres i Orchestrator for ellers kan man ikke *udvikle* og *debugge* sine processer. Det er muligt man skal kontakte UiPath for at aktivere sin gratis DEV-licens.

Man kan lave en testrobot som indeholder alle de vigtigste/ønskede funktioner som man vil teste. Testrobotten kan indeholde særlig sensitiv kode eller kode som erfaringsmæssigt fejler ofte.

Ved opgradering til næste version afvikles robotten og man registrer om noget går galt. Dette fikses og beskrives og man bliver ved indtil robotten afvikler fejlfrit. Med dette indblik i evt. fejl tilrettes eksisterende robotter.

Undgå at benyt den indbyggede "Opdater-nu" funktion i Studio da man svært kan rulle tilbage i tilfælde af fejlende kode.

Saml et bibliotek af versioner, så man kan rulle frem og tilbage i forhold til kode der fejler.

Blue Prism
DEV og TEST kan evt. angives anderledes. Dette udbygges når Statens It har gennemført pilotforløb med Blue Prism.

Produktion
Produktionsmiljøet er magen til udviklingsmiljøet. Dette er et bevidst valg og gør at robotten ser det samme som ved udvikling.

UiPath
Robotter i PROD angives i Orchestratoren ved at tildele dem en PROD-licens. Når de er i PROD, kan man ikke længere debugge dem og alle logfiler placeres standardmæssigt i Orchestrator.

Der bør være flere servere til ORCH idet opgradering til nyeste softwareversion rammer alle tenants. Det er ikke sikkert at alle robotter er parate på opgraderings-tidspunktet. Det vil være ønskeligt, at en anden ORCH får den nye version og så kan man gradvist overføre robotterne hertil indtil alle er overflyttet (og evt. tilret-tet). Bag efter man lukkes den gamle eller genbruge denne til næste version.

Prioritering
Det kræver, at der i organisationen er en grundlæggende respekt for at miljøerne prioriteres med hensyn til opdateringer, fejlsøgning og rettelser.

Og at det accepteres, at udviklerne prioriterer sådan her:

1. Driftsproblemer
2. Problemer eller opdateringer i test/udviklingsmiljøer
3. Udviklings-/projektarbejde

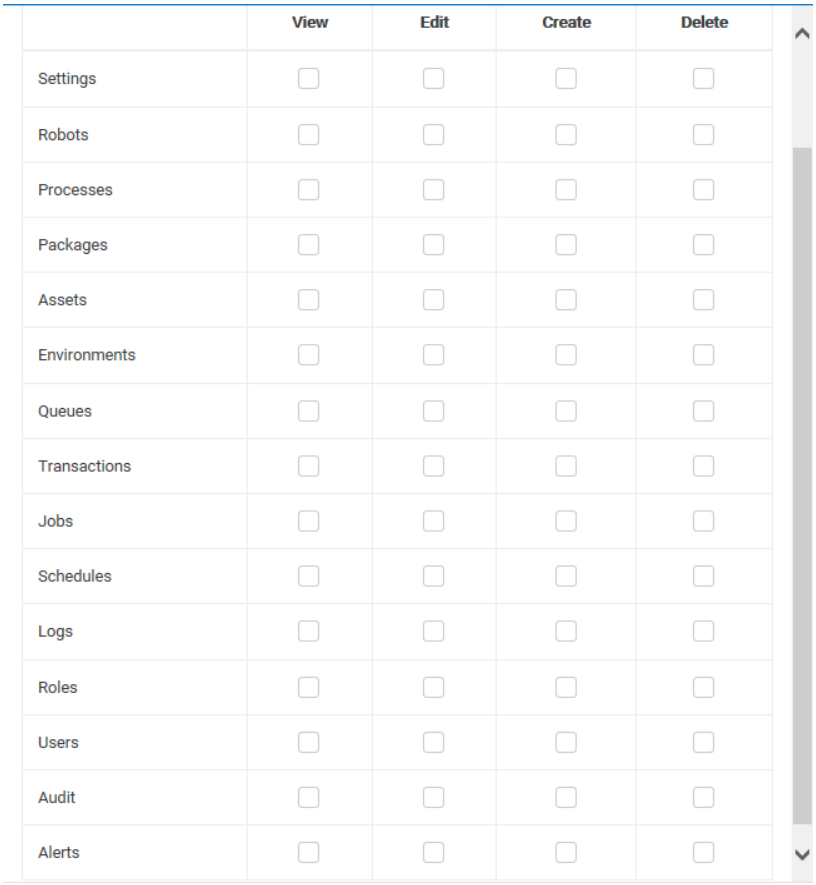
Hvis denne prioritering ikke efterleves vil selv en lille test af en rettelse hurtigt blive uforholdsmæssig stor og dyr.

Roller

RIA-platformen opererer med roller og profiler. De er forskellige i de underliggende RPA-teknologier, så de beskrives per teknologi.

UiPath

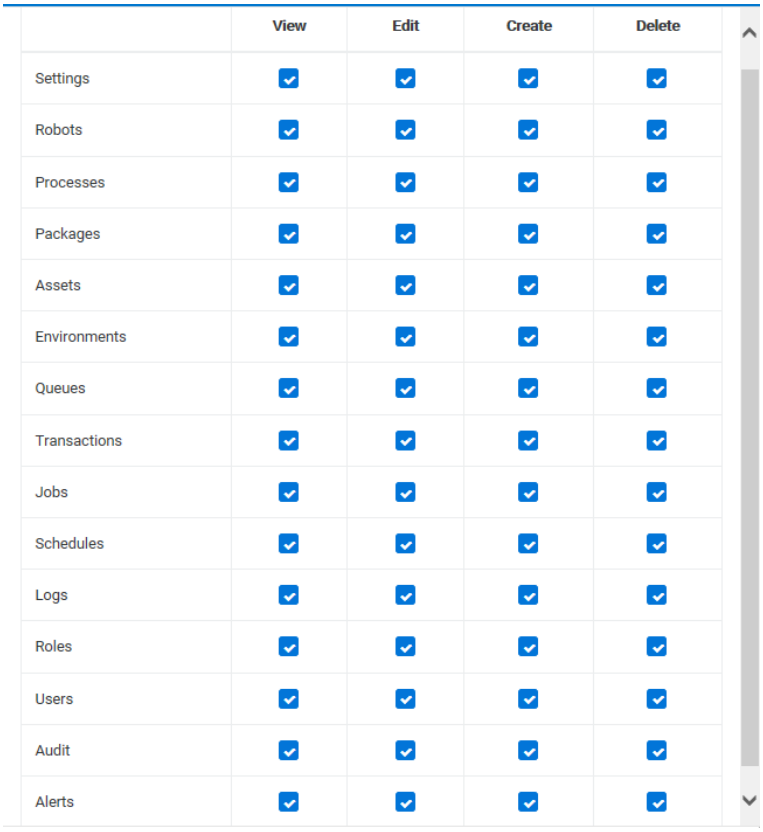
UiPath i *Multi Tenancy Mode* definerer de følgende settings:



	View	Edit	Create	Delete
Settings	☐	☐	☐	☐
Robots	☐	☐	☐	☐
Processes	☐	☐	☐	☐
Packages	☐	☐	☐	☐
Assets	☐	☐	☐	☐
Environments	☐	☐	☐	☐
Queues	☐	☐	☐	☐
Transactions	☐	☐	☐	☐
Jobs	☐	☐	☐	☐
Schedules	☐	☐	☐	☐
Logs	☐	☐	☐	☐
Roles	☐	☐	☐	☐
Users	☐	☐	☐	☐
Audit	☐	☐	☐	☐
Alerts	☐	☐	☐	☐

Der defineres automatisk to roller per tenant: *Admin* og *Robots*, og disse kan ikke slettes.

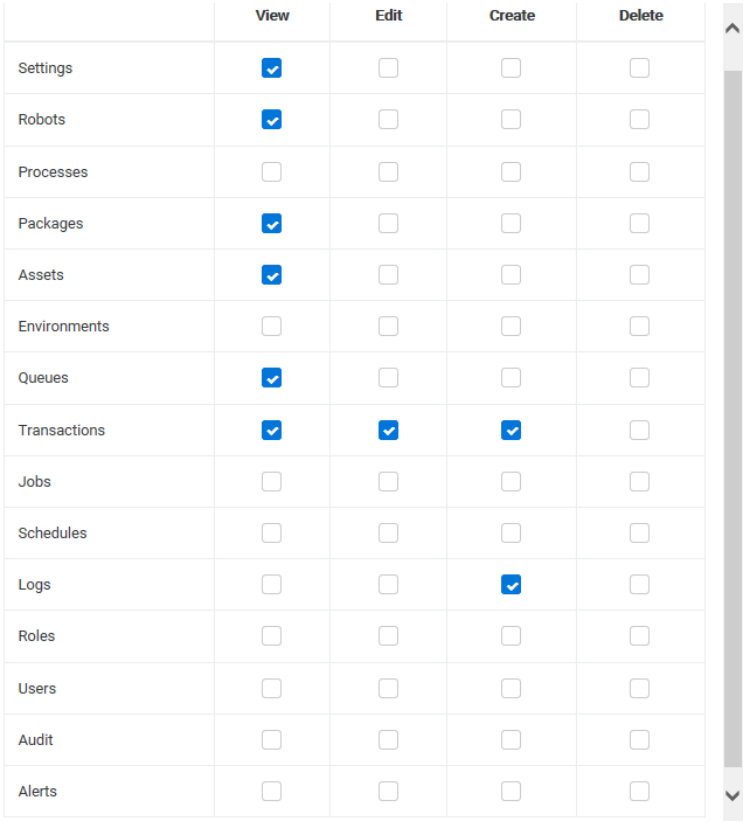
Administratorrollen



	View	Edit	Create	Delete
Settings	☑	☑	☑	☑
Robots	☑	☑	☑	☑
Processes	☑	☑	☑	☑
Packages	☑	☑	☑	☑
Assets	☑	☑	☑	☑
Environments	☑	☑	☑	☑
Queues	☑	☑	☑	☑
Transactions	☑	☑	☑	☑
Jobs	☑	☑	☑	☑
Schedules	☑	☑	☑	☑
Logs	☑	☑	☑	☑
Roles	☑	☑	☑	☑
Users	☑	☑	☑	☑
Audit	☑	☑	☑	☑
Alerts	☑	☑	☑	☑

Administratoren kan *alt*.

Robotrollen



Bemærk, at en robot skal kunne aflæse (*view*) information i mange områder for at kunne eksekvere sine workflows. Rettigheder til *Logs* sikrer at logfiler oprettes og opdateres under kørslen.

Område	Beskrivelse
Settings	Giver adgang til Settingssiden. Det er siden hvor generelle applikationsindstillinger er tilgængelige, fx SMTP-server, FROM-email-adresse, tidszone etc.
Robots	Giver adgang til Robotsiden. Siden håndterer robotter og environments.
Processes	Giver adgang til Processiden. Siden håndterer de aktive processer (packages) som robotterne kan eksekvere for den aktuelle tenant.
Packages	Giver adgang til Packagessiden. Her vises alle pakker som Orchestrator kender – også fra andre tenants.
Assets	Giver adgang til Assetssiden. Her håndteres globale værdier som robotterne kan hente og bruge i deres afvikling af pakker. Der er mulighed for TEXT, BOOL, INTEGER eller CREDENTIALS.
Environments	Giver adgang til Environmentsiden. Håndterer de "miljøer" som robotter kan afvikles under, fx TEST, PROD etc. Dette er en logisk inddeling som kombineres med pakke og robot i Schedule.

Queues	Giver adgang til arbejdskøerne (Queues). Er de datakøer som føder processer der tillader flere robottere at arbejde på samme proces.
Transactions	Giver adgang til Transactionssiden. Viser de transaktioner som foregår i køerne.
Jobs	Giver adgang til Jobssiden. Et job er en pakke som udføres af en robot (manuelt) eller via en schedule. Resultatet af kørslen vises her.
Schedules	Giver adgang til Schedulessiden. En skemalagt kørsel på et bestemt tidspunkt. Her angives proces, tidspunktet for kørsel og hvilke(n) robot(ter) som skal udføre kørslen.
Logs	Giver adgang til Loggen. Registrerer de enkelte steps, fejl, logbeskeder, advarsler og andre trin for hver process og schdule.
Roles	Giver adgang til Rolessiden. Håndterer roller som tildeles de enkelte brugere. Der er mange flueben, så man kan styre adgangen temmelig præcist for de enkelte roller. Der oprettes altid **Administrator** (som kan alt) og **Robot**.
Users	Giver adgang til brugeradministrationen for den aktuelle tenant. Hver bruger som må logge på Orchestrator oprettes her og tildeles en eller flere roller. Der findes altid **admin**. Resten opretter man selv efter behov.
Audit	Giver adgang til Auditsiden i administratormenuen. Audit er en oversigt over hvad der sker i administrationen af OPRCH, altså når man ændrer i Users.
Alerts	Giver adgang til Alertsiden. Siden viser hvilke notifikationer (primært fejl) der er opstået i systemet.

Det er muligt at bygge egne roller som passer på ens egen opsætning. Til at gøre dette benyttes administratorrollen, da det frarådes at give andre tilladelse til dette. Orchestrator tilbyder desuden gendannelse af adgangskoder via. skulle koden mistes. I sidste tilfælde, kan Statens It nulstille adgangskode.

Med de UiPath-roller, som Statens It har oprettet, udføres følgende funktioner:

Rolle	Beskrivelse
Developer	Udvikling og debug. De har adgang til ressourcer som log, køer og læseadgnag til robotter og assets.
Supervisor	Står for den daglige drift af kundens tenant. Har har adgang til alt undtagen Brugere. Det er ham som håndterer robotter, køer og assets samt tildeler licenser og sætter pakker i produktion.
Administrator	Har adgang til alt og benyttes kun i nødstilfælde eller ved upload af licensfil til tenant.

Supervisorrollen
Statens It leverer en mini-admin rolle som kan alt undtagen at redigere brugere. Den var oprindeligt tænkt som en erstatning for den rigtige adminrolle. Nu da alle kunder er admin i eget hus, kan rollen anvendes i det daglige for at varetage funktioner som:

- Oprettelse og håndtering af robotter
- Tildeling af licens(typ)er
- Oprettelse og håndtering af køer og transaktioner
- Oprettelse og hpndtering af schedules og jobs
- Tildeling og håndtering af processer (pakker)

Den rigtige adminrolle anvende kun ved systemarbejde, håndtering af brugere og auditering.

Developerrollen
Denne rolle er tiltænkt udviklerne som gerne må tilgå Orchestrator men ikke må håndtere PROD-miljøets ressourcer. De kan håndtere køer, transaktioner og logs. Alle andre område må de kun "se" ikke udføre opgaver i. Dette er i tråd med *Segregation of Duties* som bl.a. siger at en udvikler ikke må sætte processer i PROD. Derfor Superviser og Developer rollerne.

Administratorrollen
Som enhver anden adminsitrator så har man uindskrænket adgang og rettigheder til alle dele af sin tenant. Det betyder også at man kan lave rav i den, og derfor bør rollen begrænses til:

- Oprette brugerroller/konti
- Auditere brugere
- Uploade licensfil

Center of Excellence
Statens It benytter det ikke, men ønsker man selv at gå i gang med *UiPath Center of Excellence*, så kan man hente mere info herom på UiPaths hjemmeside her: https://www.uipath.com/center-of-excellence.

Kort fortalt går det ud på at lave en decentral enhed som består af følgende roller:

Rolle	Beskrivelse
RPA Sponsor	En person fra forretningen der oprettes teknologien som en virksomheds strategiprioritet og beskrive/identificere forretningsressourcer.
RPA Champion	En person som driver indføring af RPA i virksomheden og vogter over RPA løsningen
RPA Change Manager	En person som er vigtig for indføring af RPA I virksomheden og som oprettes Change- og Kommunikationsplaner så de passer med projektets leverancer.
RPA Business Analyst	Ekspert i forretningens processer og er ansvarlige for definition og beskrivelse af dem.
RPA Developer	Designer, udvikler og tester automationen.
RPA Supervisor	Håndterer robotarbejdsstyrken og ressourcer fra Orchestrator.
RPA Service Support	Første linje support når automationen er sat i drift

Rollerne *Solution Architect* og *Infrastructure Engineer* varetages af Statens It.

Ideen med et CoE er at man har decentral viden om forretningens processer og (muligvis) central udvikling og drift. Dette giver en god automation idet viden inkorporeres direkte fra forretningen og udvikles af dem som er gode til forretning.

Afvikling

Afvikling af en robotproces sker på en VDI robotklient og altid under en j-konto.

UiPath

Robotklienter skal tilmeldes en Orchestrator. Tilmelding kræver:

- PC-navn
- AD-konto + kodeord (robotten åbner en windows session med dette)
- URL til Orchestrator (https://sit-rpa001t.prod.sitad.dk) (sti skal passe med udstillet certifikat)

Orchestrator svarer med en unik GUID som skal indsættes på robotklienten (kræver lokal admin rettigheder). Når klient er indmeldt på Orchestrator viser serveren det med et ikon (grøn ring) og begynde at sende pakker til den.

Den AD-konto man benytter ved tilmelding, er den konto som afvikler pakker sendt til klienten. Det kan give udfordringer hvis kontoen ikke har de fornødne rettigheder til de ressourcer som workflowet tilgår.

Kontoen der skal benyttes skal være af typen J-bruger, som er en RPA serviceaccount.

UiPath Orchestrator vil afvikle en process ved at henvende sig til robotklienten, logge ind med den registrerede credentials og sætte afviklingen i gang.

Robotklienten tjekker først, at seneste version af pakken ligger parat på den lokale klient. Gør den ikke, hentes den ned fra ORCH. Herefter startes processen. Efter endt proces logges af klienten og ORCH opdaterer jobbet i sin liste.

Hvis man tilfældigvis er på klienten (logget ind med den rette konto) så afvikles processen mens man ser på og kan følge med i alle trin. Efter processen ender *forbliver* man logget på klienten (der logges altså ikke ud).

J-konto

En J-bruger kan i fagsystemer og i Statens It's Active Directory adskilles fra almindelige 'menneskelige' brugere (B-brugere) og konsulenter (X-brugere).

- En J-bruger har i udgangspunktet ikke nogen rettigheder når den leveres, da udviklere tildeler rettigheder 'nedefra' og sørger for at den enkelte J-bruger har så få rettigheder, som muligt for at sikre at princippet om 'separation of duties' overholdet.
- Kunden og Statens It kan i samarbejde tildele en J-bruger rettigheder i dialog med ejeren af det relevante fagsystem, hvor robotten skal udføre handlinger.

- Rent teknisk er J-brugeren en form for systemkonto med de forskelle at robotbrugeren ikke skal skifte kodeord, rammes af færre gruppepolitikker end almindelige brugere, men derfor har højere krav til kompleksiteten af adgangskoden til brugeren (min. 15 tegn).
  (indsæt illustration af rettigheder ift. almindelige brugere)
- Statens It anbefaler at credentials til J-brugeren gemmes i krypteret form i f.eks. Orchestratoren (UiPath), frem for i klar tekst.

Platform

Hver RPA-teknologi har sin egen måde at fungere på. Her beskrives de i hver deres afsnit.

UiPath

Information om RIA-UiPath:

RIA tenant login	https://sit-rpa001t.prod.sitad.dk/
UiPaths hjemmeside	https://www.uipath.com/
UiPath online dokumentation	https://sit-rpa001t.prod.sitad.dk/
UiPath Forum	https://forum.uipath.com/
UiPath Tech Support & Activation	https://www.uipath.com/contact-technical-and-activations
UiPath tutorials	https://www.uipath.com/resources

Udvikling

Introduktion
De følgende kapitler er ret tekniske og henvender sig primært til udviklere.

Da RIA-platformen er en hotelløsning, hvor hver kunde har sit eget værelse, men alle bor i samme hus, så er det vigtigt at visse regler omkring udvikling overholdes. Styringsserveren er fælles for alle, og kan godt bliver overbelastet hvis man ikke koder sine robotprocesser rigtigt. I det følgende giver vi nogle retningslinjer for god skik og naboskab.

Standarder
Der er nogle standardmetode og –funktioner som benyttes i SIT. Mange af dem er valgfrie, men hvis de implementeres så bør det gøres efter den følgende beskrivelse.

Ansvar
Når robotten sættes i PROD er ansvaret for dens eksistens delt i to:

Procesejeren har ansvaret for at robotten udfører det job som svarer til den proces den er skabt til. Dette omfatter både *hvad* robotten gør og *at* den gør noget. Overvågning af robottens arbejder kan kun evalueres af procesejeren da han kender processen og ved hvornår der er noget galt.

Robotejeren er typisk den person som har robotten (j-kontoen) som en virtuel medarbejder. Han er ansvarlig for at robotten kører (grøn i joboversigten). Hans opgave er også at tildele resourcer (pakker, VDI-klienter og håndtere køer og logfiler).

Hvis robotten fejler, vil **procesejeren** typisk bliver opmærksom på at sagerne hober sig op. Han vil kontakte **robotejeren**, som ser efter om robotten kører eller er gå i dvale/uendelig loop: da vil man typisk genstarte robotten. I tilfælde af alvorlige fejl, tager robotejer robotten offline og kontakter procesejer der efterfølgende kontakter sin udvikler. Udvikleren udbedrer fejlen og robotten sætter igen online.

Robotkontoen (j-konto) har i sit Managerfelt angivet hvem der ejer denne, så dette kan man relative nemt slå op.

Procesejer vil være kendt. Alternativt står det i SDD.

Credentials
De credentials som workflowet og robotten skal benytte under afviklingen skal placeres på Orchestrator som *Assets*. Udvikles der initialt uden Orchestrator skal disse lægges i legitimationsoplysninger (credential manager).

Deployment

Deployment af procespakke til robot
Beskrivelse af sikring for, at robotpakke er deployet korrekt fra Studio/Repositorie og ud på Orchestrator.

UiPath understøtter Apache SubVersion (gratis) eller Microsoft Teeam Foundation Server (dyrt tilkøb). Eller man benytter en af de mange versioneringsværktøjer (online eller on premise) hvor man i større eller mindre grad kan scripte sig ud af tjek-ind og –ud af kode. Der er en god ide at benytte et checksumsværktøj, så man overfor Revisionen kan dokumentere at det man sætte i drift er uændret fra det man har udviklet.

Best practice på RPA-platformen

Her følger en række anbefalinger og *best practice* omkring RPA-udvikling.

Udvikling
Generelt gælder at koden er på engelsk for at undgå at danske tegn skaber problemer: variabler, feltnavne, funktionsnavne etc.

Aktiviteter kan valgfrit være på dansk eller engelsk, da de ikke "kaldes" af koden. Det samme gælder kommentarer. Undgå dog at skifte sprog for ofte.

Projekthierarki
Opret et hierarki med følgende struktur:

- *Files*
  - *Functions* (funktionalitet)
  - *Components* (business rules – mindst to invokes af anden funktion)
  - *Subprocesses* (samling af componenter – mindst et invoke)
  - *Statemachine*

I hver undermappe placeres filer med samme funktionalitet. Funktioner er selvstændige rutiner som kan kaldes (evt. med parametre). Forretningslogik og større komponenter placeres i *Components*.

Navngivning, kommentarer og logning
Funktioner navngives efter dette princip:

- Func_'System name' - 'Purpose'
  - Fx: Func_SAP - Launch and Login
  - Prefix: Func, Comp, Sub
- Funktioner indeholde ikke Business Rules (forretningslogik): de returnerer kun simple værdier (fx opslag, antal, referencer) og genbruges globalt

Aktiviteter navngives efter samme princip:

- 'Activity' – 'Purpose'
- Fx: 'Type' – 'USERNAME into login'

Kommentering (*Annotations* – ikke *Comments*, højreklik på aktiviteten) bør som minimum omfatte:

- Logs
- ifs

- Assigns
- Den første/øverste *Sequence* bør fortælle hvad blokken gør (dansk eller engelsk)

Variabler og argumenter/parametre navngives efter dette princip:

- Variabler skrives i *CamelCase* (småt begyndelsesbogstav): *userName*
- Argumenter skrives i *PasalCase* (stort begyndelsesbogstav): *Func_login(UserName, Password)*
- Lokale variabler præfikses med "_", fx *_loopCount*
- Så vidt muligt, defineres variabler på forhånd, gerne med en default/dummyværdi 0 (tal) eller "¤" (streng). Undgå tomme strenge (*String.Empty*)

Logning (debug: trace, information: til Orchestrator):

- Angiv hvornår en komponent, subprocess og evt. funktion starter og slutter, udskrive gerne "---" omkring første/sidste linje. Så er det nemmere at skimme logs.

Variabler vs argumenter

Parsing af variable til andre datatyper – beskriv gerne hvordan forskellige typer data skal 'oversættes'. Det kan være meget afhængigt af sprogpakker osv. Datoformatet er det indlysende eksempel eller rækkefølgen i arrays.

Programming Customs

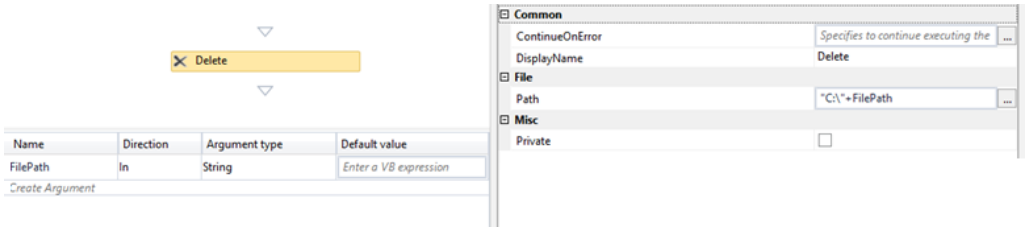
- *Element exists* benyttes altid inden man tilgår et element
- **Busines Rules** kodes i komponenter og håndterer deres egne fejl

Exception handling (Business Rule Exception VS Application Exceptions).

- **Busines Rule Exception** er de "kendte"/"tilladte" fejl som kan forekomme og som processen kan klare uden at gå i stå.
- **Application Exception** er runtime fejl som opstår fordi man ikke kunne forudse dem.
- Fejl skal så vidt muligt fanges i komponenten eller funktionen.
- Ved **Throw** går den til næste **TryCatch**?

Caveat!

Tjek variabler inden de overføres til vigtige/"farlige" funktioner og aktiviteter. Eksemplet nedenfor viser en sletning af fil. Der overføres korrekt **FilePath** til aktiviteten. Hvis argumentet er blankt (*String.Empty*) så vil sletning i sin aktuelle udformning slette hele C-drevets rod, som nok ikke er meningen.



Drev og filnavne generelt
Funktionen **Path Exists** kan ikke finde ud af at tilgå netværksdrev med bogstav (fx F:\Rapportering), den skal have fuld UNC (\\server\share\Rapportering).

Når man benytter netdrev, så være opmærksom på at Windows skal have tid til at forbinde sig til disse eksterne kilder inden brug. Det tager ca. 1 min for Windows at forbinde sig til UNC-stier efter login. Trådløse forbindelse ca. det dobbelte.

Danske tegn i prompts / UNC-stier og command line skal "oversættes":

- Original: øv hvor går du langsomt, det er ærgerligt
- Escape: ›v hvor g┼r du langsomt, det er ‘rgerligt

Prøv evt. med: "chcp 850" i *command line* for at undgå problemer.

Outlook
Mails via Outlookskabeloner oprettes og sendes som forventet: mailen bliver lagt i *udbakken* og den sendes først når Outlook *er online*. Det betyder, at Outlook skal startes *hver gang* eller *være tændt* når skabelonen afsendes - ellers sker der ikke noget før Outlook startes op næste gang (*deferred mail*).

Overvejelser ved udvikling i PROD
Ikke alle systemer har et DEV- eller TEST-miljø og det betyder at man arbejder direkte i PROD med live data! For at minimere datatab og potentielle farlige situationer, så er her nogle punkter til overvejelse:

- Arbejd sideløbende med forretningen (mange 'har jeg forstået det rigtigt' spørgsmål) så man er helt sikker på at man ramme de rigtige elementer og data.
- Indbyg default/testværdier i koden således at man ikke "kan komme til" at udføre noget med live data uden først at switche til PROD-tilstand.
- Benyt i fagsystemet, man koder op imod, en testsag, testbruger, testrolle eller benyt kun nogle få aftalte felter, som forhindrer at man kommer til at ramme andre (live) sager. Læg evt. ind i koden at man i TEST-tilstand altid benytter disse indstillinger.

- Udfør evt. alle funktioner op til den endelige submit/gem men undlad at trykke på knappen/aktivere funktionen. Dette er trinet inden *point of no return* og man kan stadig nå at rette evt. fejl.
- Indbyg en *environment switch*-funktion i Studio så man i toppen af koden kan indikere om man kører i TEST eller PROD. Testtilstanden kunne aktivere nogle af de ovennævnte punkter. Lad ev.t robotten smage på dette og selv finde ud af i hvilket miljø den arbejder i.
- *Point of no return* er når man tester sin robot på live data (evt. går i PROD) for første gang.
    - Sørg for at være helt sikker på at, alt er på plads
    - aftal evt. en fall-back metode i tilfælde af fejl
    - kør altid én iteration igennem ad gangen i testfasen. Loop senere når alt er OK.

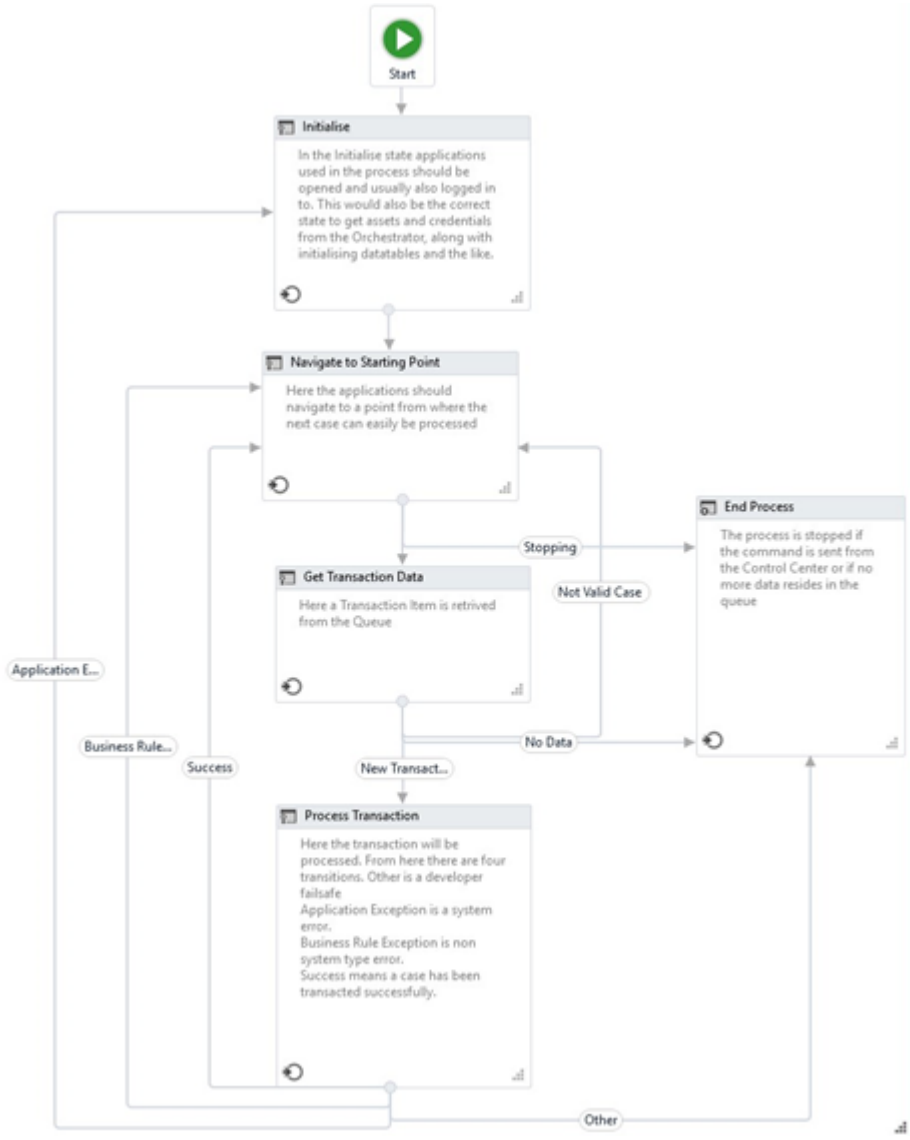
Hvis man tilfældigvis er på klienten (logget ind med den rette konto) så afvikles processen mens man ser på og kan følge med i alle trin. Efter processen ender *forbliver* man logget på klienten (der logges altså ikke ud).

Dette kan man udnytte i TEST/DEBUG. Log på klienten via RDP med den konto der skal afvikle processen. Start processen manuelt fra *UiPath Robot dialogen* i proceslinjen (ved uret). Du kan nu følge med i hvad robotten laver og hvor den evt. fejler. Hvis den fejler kan du trykke F12 (*Cancel Execution*) og med det samme starte UiPath Studio op. Du kan dog ikke *redigere* i den aktuelle proces (det er ikke projektet men nuGet-pakken som udføres på klienten), men du kan *aflæse Selectorer* på uventede dialogvinduer og andre elementer. Disse kan du så kopiere via RDP til dit eget udviklingsmiljø og tilrette processen. Upload en ny pakke til ORCH og prøv igen.

Da **Assets** ikke kan ændres af robotter i *runtime*, så benyt evt. en særlig kø til kommunikation mellem processer eller dynamiske globale variabler.

Robotskelet – brug evt. state machines

Vi anbefaler at man bygger sine processer op i state machine. Nedenfor er et eksempel fra Deloitte.



Det anbefales at følge en af disse måder at opbygge robotter. Det sikrer at Orchestratoren kan afvikle (og afrapportere på) robotten og samtidig er workflows i stand til at afvikle faste bibliotekskald (fx tidsforbrug og filhåndtering).

1) Brug *workflows* som default og markér de vigtigste forgreninger.
2) Brug *sequence*-gruppen (gerne indlejrede) til at gruppere funktioner og aktioner der er logisk beslægtede.
3) Brug *Target*-egenskaben til at holde fokus på applikation og elementer. Især når der benyttes serier af Hotkeys.
4) Anvend *Log Message* i stedet for *Write Line* til at udskrive værdier og kommentarer til logfilen. Benyt disse værdier:

a) *Trace* – Normal tekst
b) *Info* – Information om fremhævet/særlig aktivitet, fx fil- og skabelonnavn, emnenummer ved iteration (loops), fremhæve start/slut af funktion.
c) *Warn* – Besked ved anomalier: Forkert indtastet dato som justeres, brug af defaultværdi i stedet for forventet input.
d) *Error* – Ved deciderede fejl: manglende resourcer så som manglende fil, ingen input, forkert dato etc.
e) *Fatal* – Ved ikke-planlagt programafbrydelse, filnavnet på en skabelon mangler og applikationen kan ikke fortsætte.

5) Navngivning efter Deloittes anbefaling:
   a) alle variabler starter med småt
   b) argumenter starter med stort
   c) al programmering på engelsk
6) Funktioner som inkluderes skal ligge i samme folder som hovedprogram og skal være navngivet *sub*, globale variabler til brug i disse prefikses med *sub.*
7) *Should stop* angiver et sted i workflowet hvor robotten kan afbryde når den modtager et *cancel/terminate* signal fra Orchestrator. Dette giver samtidig en escape-mulighed ved loopkørsler. Tjekket bør lægges så robotten enten:
   a) Ikke (endnu) foretager sig noget (fx i starten af workflowet)
   b) Har foretaget noget, som kan gentages uden problem
   c) Eller brancher til en rutine som ruller aktioner tilbage som er foretaget indtil dette tidspunkt. Dette er ikke altid muligt, så brug #a eller #b
8) Benyt *Invoke workflow File* hvor noget kan genbruges. Men vær dog opmærksom på om/at dette ikke komplicerer eller forlænger den totale afvikling af processen.

Bilag 1 - Begreber

Aktivitet – en handling i en automatiseret proces, fx åbn Excel, skriv til log, send mail etc.

Back-Office robot (BOR, unattended robot) – Selvstændig afvikling af robot-jobs fra Orchestrator. Denne robottype er velegnet til selvstændigt at udføre gentagende, regelbaseret og ensformigt arbejde.

Front-Office robot (FOR, attended robot) – En robot som startes af brugeren og arbejder i den lokale kontekst af brugerens miljø. Det kan fx være en færdigprogrammeret hjælp i Navision som gør de 15 mest tidskrævende (eller fejlbehæftede) aktioner for brugeren der så efterfølgende godkender eller fortsætter den manuelle arbejdsproces. Denne type robot er velegnet til processer som kræver en menneskelig vurdering eller ikke er strengt regelbaseret.

Job – Eksekvering af et workflow enten lokalt (FOR) eller fra Orchestratorserveren (BOR). Overvågning og logning foretages af Orchestratoren (BOR) eller af den lokale arbejdsstation (FOR).

Lifecycle – En beskrivelse af workflow fra vugge til grav. Hos UiPath deler man livscyklussen op i fire dele:

Design	Beskriver hvert enkelt trin af processen som robotten skal udføre
Development	Workflowet oprettes på baggrund af designbeskrivelsen
Testing	Test, fejlretning og optimering af workflowet, gerne i testmiljø
Production	Det færdige workflow sættes i produktion, oftest fra Orchestratoren og den daglige drift overvåges via dashboard og logfiler.

Ydermere arbejdes med en tretrinstilgang:

PoC	Proof of concept – hvor man viser om en arbejdsgang kan automatiseres og tilpasses til en robot

Pilot	Testkørsel af workflow i testmiljø. Her skal workflowet typisk justeres lidt og gøres robust. Hvis robotten skal kunne skaleres så er det typisk her man finder ud af at tilføje/justere Queues.
Roll-Out	Det færdige workflow introduceres til i produktionsmiljøet. De første kørsler er typiske manuelle indtil man er sikker på at alt kører som det skal. Herefter er går man over til almindelig drift (og overvågning).

Orchestrator – (UiPath) Det er en browserbaseret serverapplikation der anvendes som kommandocenter til planlægning, styring og overvågning af robotter.

Proces – det samme som workflow. Benyttes mest om den menneskelige aktivitet versus det automatiserede slutprodukt (workflow)

Queues – Når et workflow skal kunne afvikles af flere robotter, så skal man tilpasse det ved at benytte en køsturktur. I stedet for at hver robot udfører samme arbejde på potentielt samme opgave/data, så anvendes én robot til at hente information for alle opgaver og placerer dem ind i en kø. De andre robotter kan nu bede Orchestratoren om at få et job og Orchestratoren sender data fra køen ud til hver robot. Derved undgår man at 4 robotter udbetaler feriepenge til samme person fire gange. Orchestratoren sørger for, at hver robot får sin egen person til feriepengeudbetalingen. Det kræver naturligvis at workflowet er designet hertil og at man holder flere robotter parate.

Robot – Den del af UiPAth som afvikler et workflow. Termen robot benyttes ofte om den proces robotten udfører fx feriepengerobotten eller mødebookingrobotten – selv om man strengt taget mener det reelle workflow eller job.

Roller – UiPath opererer med roller for at identificere og styre de aktiviteter som udføres i lifecyklussen.

Developer	Personen der udarbejder workflows
Monitor	Personen der overvåger driften i Orchestrator og Kibana Dashboard.

UiPath Studio – værktøjet som opretter workflows via en visuel brugerflade.

Workflow – En samling aktiviteter som repræsenterer den automatiserede proces. En række instruktioner som fortæller robotten hvordan den skal udføre sit arbejde. I UiPath udarbejdes disse i *UiPath Studio* og gemmes som XAML-filer. De publiceres til ORCH hvorfra de afvikles på BOR eller FOR.