STS, KMD I2 og STIL

Installation og konfiguration

**Version**: 1.3.0

**Date**: 26.03.2021

**Author**: BSG

Indhold

[1 Indledning 4](#_Toc67662189)

[1.1 Binære artifakter 4](#_Toc67662190)

[1.2 Licens og ejerskab 4](#_Toc67662191)

[1.3 Overblik over deployment 4](#_Toc67662192)

[1.4 Docker konfiguration 4](#_Toc67662193)

[2 Rolle-register opsætning (AD) 5](#_Toc67662194)

[2.1 Windows Service 5](#_Toc67662195)

[2.1.1 Konfigurationsfil 5](#_Toc67662196)

[2.1.2 Afvikling 6](#_Toc67662197)

[2.1.3 Log fil 6](#_Toc67662198)

[2.2 Docker container 6](#_Toc67662199)

[2.2.1 Konfiguration 7](#_Toc67662200)

[2.3 Test af korrekt opsætning 8](#_Toc67662201)

[3 Rolle-register opsætning (OS2rollekatalog) 8](#_Toc67662202)

[3.1 OS2rollekatalog 8](#_Toc67662203)

[3.2 Docker Container 10](#_Toc67662204)

[3.2.1 Konfiguration 10](#_Toc67662205)

[3.3 Test af korrekt opsætning 12](#_Toc67662206)

[4 KMD I2 opsætning 12](#_Toc67662207)

[4.1 Docker Container 12](#_Toc67662208)

[4.1.1 Konfiguration 12](#_Toc67662209)

[4.2 Test af korrekt opsætning 14](#_Toc67662210)

[4.3 Yderligere API’er 15](#_Toc67662211)

[5 OS2sync opsætning 15](#_Toc67662212)

[5.1 Stubbet udgave 15](#_Toc67662213)

[5.1.1 Konfiguration 16](#_Toc67662214)

[5.2 Test af korrekt opsætning 17](#_Toc67662215)

[6 Opsætning af integrationsflow 17](#_Toc67662216)

[6.1 Docker Container 17](#_Toc67662217)

[6.1.1 Konfiguration 17](#_Toc67662218)

[6.2 Test af korrekt opsætning 19](#_Toc67662219)

[7 Overvågning 19](#_Toc67662220)

[7.1 Overvågning af applikationens ”liveliness” 19](#_Toc67662221)

[7.2 Overvågning af applikations log 19](#_Toc67662222)

# Indledning

Nærværende dokument er en samlet vejledning til opsætning af KMD I2 integrationsløsningen. Dokumentet forholder sig alene til opsætningen af integrationen, og ikke til den egentlige funktionalitet af løsningen. For dette henvises til løsningsbeskrivelsen.

## Binære artifakter

Det er muligt at hente kildekoden til løsningen og selv bygge sine egne binære artefakter til deployment, men der er også publiceret egentlige binære som kan anvendes.

Kildekoden kan hentes her

<https://github.com/orgs/OS2services/dashboard>

De binære artefakter er publiceret hhv til Dockerhub

<https://hub.docker.com/orgs/os2services>

og i forhold til AD integrationen, som en Installer her

<https://github.com/OS2services/RolesFromAD/tree/master/releases>

## Licens og ejerskab

Løsningen er publiceret som Open Source under MPL 2.0 licensen, og er ejet af OS2 samarbejdet. Løsningen kan frit anvendes under de vilkår som MPL 2.0 licensen angiver.

## Overblik over deployment

I forbindelse med deployment af løsningen skal man tage stilling til hvilken kilde der skal anvendes for roller. Hvis man anvender OS2rollekatalog som rolle/rettighedsstyringskomponent, vil det være naturligt at anvende integrationen til denne, ellers kan man med fordel anvende AD integrationen. Det giver ikke mening at anvende begge integrationer samtidig.

Det betyder at man skal vælge at sammensætte 3 integrationer med det egentlige data-udvekslingsflow

* En integration til et institutions-register (pt understøttes kun KMD I2)
* En integration til et rolle-register (pt understøttes AD og OS2rollekatalog)
* En integration til et organisationssystem (pt understøttes kun FK Organisation)

## Docker konfiguration

I de enkelte afsnit nedenfor, er der angivet hvordan man opsætter konfigurationen af de enkelte docker containere. For dem alle gælder der at man skal bruge miljøvariable til at holde konfigurationen, hvilket i nedenstående dokumentation udføres vha docker-compose filen.

Hvis man ikke bruger docker-compose og/eller man ikke har mulighed for at angive miljøvariable, så kan man angive konfigurationen i en konfigurationsfil.

I så fald skal man mounte et volume til sin docker container der mappes ind til folderen /config, og i denne folder skal der ligge en konfigurationsfil ved navn application.properties.

Bemærk at denne fil skal være gemt i tegnsættet ISO-8859-1 ellers vil evt specialtegn (herunder ÆØÅ) ikke blive indlæst korrekt.

Filen udfyldes med de variable som man normalt ville angive som miljøvariable fx i docker-compose.

Et kort eksempel på indholdet af en sådan fil er vist nedenfor, og man kan se et fuldt eksempel i den eksempel docker konfiguration der ligger sammen med koden

configuration.cron=0 0/5 \* \* \* \*

configuration.apiKey=Test1234

configuration.roles.supportedRoles=Teacher,Pedagogue

configuration.roles.url=http://auladagtilbud-roles:8999

configuration.roles.apiKey=Test1234

# Rolle-register opsætning (AD)

## Windows Service

Hvis man vælger at anvende AD som rolle-register, skal der installeres en on-premise windows service, som kan hentes her

<https://github.com/OS2services/RolesFromAD/tree/master/releases>

Ovenstående er en EXE installer, som blot skal afvikles på den windows server hvor man ønsker at servicen installeres. Efter installationen skal konfigurations-filen tilpasses.

### Konfigurationsfil

Konfigurationsfilen findes i den folder hvor softwaren er installeret (vælges under installationen). Filen hedder

Aula Dagtilbud AD Integration.exe.config

Og indeholder en <appSettings> sektion, med den konfiguration der skal tilpasses.

Man kan som udgangspunkt beholde hele ”serilog” sektionen som den er. Den gemmer daglige logfiler (op til 30 dage) i den angive folder. Man kan ændre folder-placeringen og filens navn hvis dette ønskes. Yderligere konfiguration kan fortages af logfilen hvis dette ønskes, og her henvises til den officielle Serilog dokumentation

<https://serilog.net>

Der er 3 indstillinger som SKAL tilpasses i forbindelse med opsætningen, og disse er

**webSocketKey**

Denne værdi skal udfyldes med den nøgle der er opsat i containeren (se nedenfor). Det er samme værdi som man her har opsat i ”configuration.webSocketKey”. Denne værdi sikrer at kommunikationen mellem containeren og windows servicen er sikret.

**webSocketUrl**

Her angiver man adressen på containeren som der skal kommunikeres med. Bemærk at protokollen der anvendes er websockets, og der derfor skrives ws:// eller wss:// i stedet for de normale http:// eller https://.

Hvis man har TLS/SSL opsat foran sin container, skal man bruge wss:// som protokol, ellers ws:/

Eksempel på en korrekt værdi er

ws://192.168.1.100:8899/ws

Bemærk at uagtet IP/DNS navn og port, så skal URL’en afsluttes med /ws da dette er det konkrekte endpoint på containeren hvor den kommunikerer med websockets.

**uuidAttribute**

Denne værdi kan man som udgangspunkt udelade (slette helt fra konfigurationsfilen). Hvis man gør det, så anvendes objectGuid som UUID på brugerne i AD. Hvis man har valgt at UUID skal læses fra et andet felt, så kan man udfylde denne værdi med attributtens navn fra AD.

**adUrl/adRoot/adUsername/adPassword**

I konfigurationsfilen er der 4 yderligere indstillinger. De kan blot ignoreres eller slettes helt. De anvendes hvis man drifter windows servicen udenfor kommunens netværk, og tillader at man kan forbinde til et specifikt AD med en specifik brugerkonto.

Hvis man afvikler windows servicen på kommunens netværk, så anvendes disse indstillinger ikke.

### Afvikling

Windows servicen skal afvikles under en service konto der har lov til at læse fra AD. Servicen udlæser de felter der er konfigureret i ovenstående konfigurationsfil, og det er vigtigt at servicekontoen har læse-adgang til disse (typisk har en almindelige servicekonto læse-adgang til alle ikke-følsomme felter, så det er alene hvis man har valgt at anvende et følsom felt at dette er relevant).

Når man starter windows servicen, forsøger den at forbinde til nedenstående docker container. Hvis den ikke er kørende, så vil den løbende forsøge at gen-etablere forbindelsen indtil den kan etableres.

### Log fil

Windows servicen logger til følgende logfil, som kan bruges til at fejlsøge på evt kommunikationsproblemer

C:\Logs\AulaDagtilbud

Ovenstående er default placeringen, men man kan ændre placeringen af logfilen via konfigurationen af windows servicen.

## Docker container

Man kan hente en eksempel docker-compose.yml fil her

<https://github.com/OS2services/AulaDagtilbudIntegration/tree/master/docker>

der viser hvordan man kan opsætte denne container til brug med Docker Compose.

Nedenstående er en gennemgang af de konfigurationsmuligheder der er til containeren.

### Konfiguration

Der publiceres images til anvendense på Dockerhub her, hvor man altid kan finde den nyeste udgave

<https://hub.docker.com/orgs/os2services>

Internt kører containeren på port 8999, men man kan vælge at mappe dette til en anden port når/hvis den udstilles udenfor det netværk containeren deployes i.

I environment sektionen af docker-compose.yml filen har man mulighed for at opsætte følgende værdier

|  |  |
| --- | --- |
| **Nøgle** | **Beskrivelse** |
| configuration.enabled | Skal sættes til “true” for at slå den faktiske forretningsfunktionalitet til. Hvis den er sat til ”false”, så kalder den ikke AD, men returnerer i stedet et statisk svar (se ”configuration.testData”) |
| configuration.apiKey | Containeren udstiller et REST API som kan kaldes af de andre containere i deployment miljøet. Her angives den nøgle som skal anvendes når der foretages kald mellem containerne.  Kald foretaget uden denne nøgle vil blive afvist af denne container. |
| configuration.webSocketKey | Skal udfyldes med et hemmelig nøgle. Dette er den samme nøgle som skal anvendes i konfigurationen af on-premise windows servicen, og anvendes som signeringsnøgle til at beskytte trafikken mellem containeren og windows servicen. |
| configuration.mapping[?].identifier | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  I ”identifier” skal man udfylde navnet på rollen som den er kendt i institutionssystemet, fx   * Teacher * Pedagogue * Osv.. |
| configuration.mapping[?].groupIdentifier | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  I ”groupIdentifier” udfylder man ID på den AD gruppe som repræsenterer den rolle som er angivet i ”identifier” ovenfor.  sAMAccountName på AD gruppen kan anvendes her. |
| configuration.testData[?].uuid | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder AD. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter.  ”uuid” udfyldes med UUID på en fiktiv bruger som har alle roller. |
| configuration.testData[?].userId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder AD. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter.  ”userId” udfyldes med AD brugernavnet på en fiktiv bruger som har alle roller. |

## Test af korrekt opsætning

Når containeren er startet, så kan man teste output fra den vha curl eller lignende REST værktøj (postman er også en mulighed)

Det antages at containeren er udstillet mod localhost på port 8999 i nedenstående, og at der findes en rolle med identifier ”Teacher”.

$ curl –header "ApiKey: xxxx" <http://localhost:8999/api/whoHasRole?identifier=Teacher>

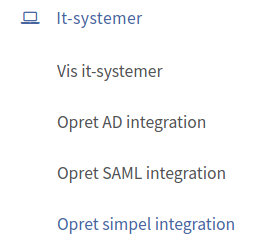
Hvis det hele er opsat korrekt, så returneres et JSON array med de brugere som har den angivne rolle

# Rolle-register opsætning (OS2rollekatalog)

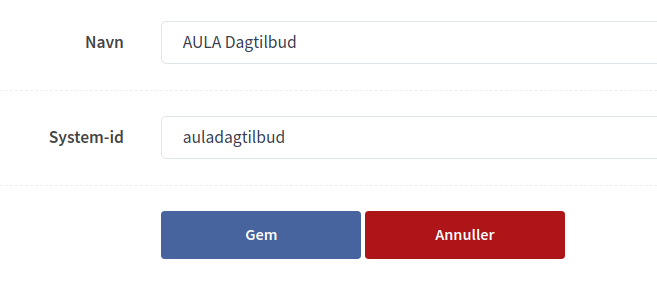
## OS2rollekatalog

Hvis man vælger at styre rolletildelingen via OS2rollekatalog, skal man først oprette et it-system inde i OS2rollekatalog til formålet.

I venstre-menuen i OS2rollekatalog, start med at klikke på ”Opret SAML integration” under it-systemer som vist nedenfor



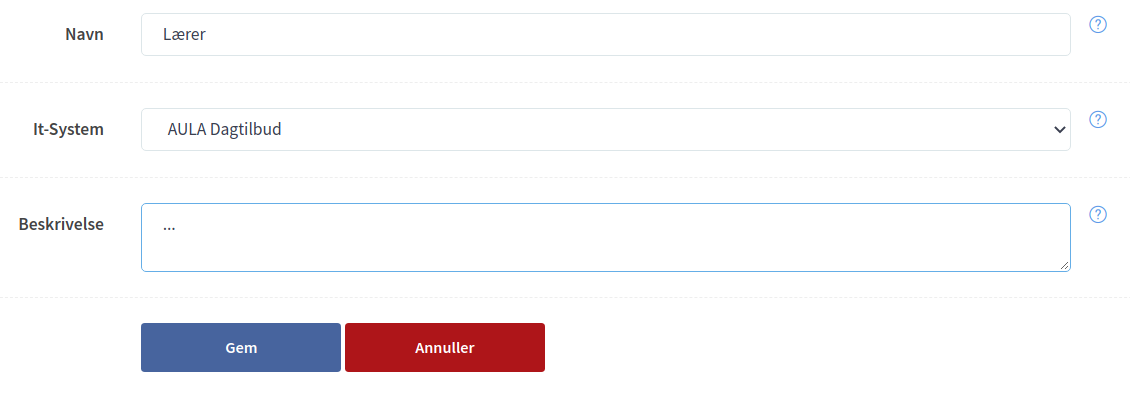
Herefter oprettes et it-system med navn og ID. Værdierne skal ikke bruges efterfølgende, men giv it-systemet et sigende navn, så det er til at finde efterfølgende.



Det er ikke nødvendigt at udfylde nogen stamdata for it-systemet på den efterfølgende side, så gå bare direkte til at oprette Jobfunktionsroller, hvilket kan gøres i venstre-menuen under Jobfunktionsroller, som vist nedenfor



For hver rolle som ønskes styret i KMD I2, oprettes en Jobfunktionsrolle. Hver rolle gives et sigende navn og der tilføjes en beskrivelse. Som it-system vælges det it-system som blev oprettet i forrige trin.



Efter en Jobfunktionsrolle er oprettet, kan man redigere indholdet af den. Dette er ikke nødvendigt, men noter det ID som Jobfunktionsrollen har fået. Dette kan aflæses i adressebasen i web-browseren som vist nedenfor



I ovenstående eksempel har vi oprettet en Jobfunktionsrolle til ”Lærer”, som har fået ID 461.

Gentage dette for hver rolle, og noter ID’et for hver rolle. Disse skal bruges i opsætningen af containeren nedenfor.

Man kan efterfølgende tildele disse Jobfunktionsroller til medarbejdere i kommunen, lige som for alle andre roller i OS2rollekatalog.

## Docker Container

Man kan hente en eksempel docker-compose.yml fil her

<https://github.com/OS2services/AulaDagtilbudIntegration/tree/master/docker>

der viser hvordan man kan opsætte denne container til brug med Docker Compose.

Nedenstående er en gennemgang af de konfigurationsmuligheder der er til containeren.

### Konfiguration

Der publiceres images til anvendense på Dockerhub her, hvor man altid kan finde den nyeste udgave

<https://hub.docker.com/orgs/os2services>

Internt kører containeren på port 8999, men man kan vælge at mappe dette til en anden port når/hvis den udstilles udenfor det netværk containeren deployes i.

I environment sektionen af docker-compose.yml filen har man mulighed for at opsætte følgende værdier

|  |  |
| --- | --- |
| **Nøgle** | **Beskrivelse** |
| configuration.enabled | Skal sættes til “true” for at slå den faktiske forretningsfunktionalitet til. Hvis den er sat til ”false”, så kalder den ikke OS2rollekatalog, men returnerer i stedet et statisk svar (se ”configuration.testData”) |
| configuration.apiKey | Containeren udstiller et REST API som kan kaldes af de andre containere i deployment miljøet. Her angives den nøgle som skal anvendes når der foretages kald mellem containerne.  Kald foretaget uden denne nøgle vil blive afvist af denne container. |
| configuration.rc.apiKey | Skal udfyldes med API nøglen til at kalde OS2rollekatalog. Denne værdi kan man udlæse fra driftsopsætningen af OS2rollekatalog. |
| configuration.rc.url | Skal udfyldes med URL’en hvor OS2rollekatalog er driftet, med /api som en del af opsætninge. Fx  <https://kommune.rollekatalog.dk/api> |
| configuration.rc.mapping[?].identifier | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  I ”identifier” skal man udfylde navnet på rollen som den er kendt i institutionssystemet, fx   * Teacher * Pedagogue * Osv.. |
| configuration.rc.mapping[?].roleIdentifier | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  I ”roleIdentifier” udfylder man ID på den tilsvarende Jobfunktionsrolle i OS2rollekatalog (de ID’er som blev noteret under oprettelsen af rollerne i forrige trin) |
| configuration.testData[?].uuid | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder AD. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter.  ”uuid” udfyldes med UUID på en fiktiv bruger som har alle roller. |
| configuration.testData[?].userId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder AD. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter.  ”userId” udfyldes med AD brugernavnet på en fiktiv bruger som har alle roller. |

## Test af korrekt opsætning

Når containeren er startet, så kan man teste output fra den vha curl eller lignende REST værktøj (postman er også en mulighed)

Det antages at containeren er udstillet mod localhost på port 8999 i nedenstående, og at der findes en rolle med identifier ”Teacher”.

$ curl –header "ApiKey: xxxx" <http://localhost:8999/api/whoHasRole?identifier=Teacher>

Hvis det hele er opsat korrekt, så returneres et JSON array med de brugere som har den angivne rolle

# KMD I2 opsætning

For at kunne integrere med KMD I2 skal der være indgået en aftale med KMD om dette. Der findes en portal til anmodninger, og en proces for aftaleindgåelse med KMD som man kan læse mere om her

<https://kmdi2prod.portal.azure-api.net/>

Der er også en testportal, som kan tilgås her til udvikling/test-formål

<https://kmdi2test.portal.azure-api.net/>

Det forudsættes at man har en aftale på plads, og man her har fået sin unikke kommunale URL og tilhørende hemmelige nøgle, da disse skal bruges til den tekniske opsætning.

## Docker Container

Man kan hente en eksempel docker-compose.yml fil her

<https://github.com/OS2services/AulaDagtilbudIntegration/tree/master/docker>

der viser hvordan man kan opsætte denne container til brug med Docker Compose.

Nedenstående er en gennemgang af de konfigurationsmuligheder der er til containeren.

### Konfiguration

Der publiceres images til anvendense på Dockerhub her, hvor man altid kan finde den nyeste udgave

<https://hub.docker.com/orgs/os2services>

Internt kører containeren på port 8997, men man kan vælge at mappe dette til en anden port når/hvis den udstilles udenfor det netværk containeren deployes i.

I environment sektionen af docker-compose.yml filen har man mulighed for at opsætte følgende værdier

|  |  |
| --- | --- |
| **Nøgle** | **Beskrivelse** |
| configuration.kmdi2.enabled | Skal sættes til “true” for at slå den faktiske forretningsfunktionalitet til. Hvis den er sat til ”false”, så kalder den ikke KMD I2, men returnerer i stedet et statisk svar (se ”configuration.testData”) |
| configuration.kmdi2.dryRun | Hvis denne værdi er sat til ”true”, så udføres der ikke opdaterende operationer mod KMD I2, men i stedet logges payload, så man kan teste hvad der ville ske hvis man faktisk udførte de opdaterende operationer. |
| configuration.apiKey | Containeren udstiller et REST API som kan kaldes af de andre containere i deployment miljøet. Her angives den nøgle som skal anvendes når der foretages kald mellem containerne.  Kald foretaget uden denne nøgle vil blive afvist af denne container. |
| configuration.allowedRoles | Man kan opsætte en restriktion på hvilke roller som integrationen skal vedligeholde. Som udgangspunkt vedligeholder integrationen alle roller, men hvis man kun ønsker at integrationen skal vedligeholde enkelte roller, så kan man udfylde denne setting. Man angiver en række af roller med komma mellem (uden mellem).  Eksempel herpå kunne være værdien  ”Teacher,Pedagogue,Substitute”  Udelad indstillingen helt hvis man ønske default opførsel. |
| configuration.kmdi2.url | Skal udfyldes med URL’en hvor KMD I2 kan kaldes. Et eksempel på en lovlig URL mod testmiljøet er (formodentligt indeholder alle kommuners URL’er deres kommunekode som det sidste i adressen)  <https://kmdi2test.azure-api.net/ww-employee/api/v1/municipalities/751> |
| configuration.kmdi2.apiKey | Denne skal udfyldes med den hemmelige nøgle som er modtaget af KMD ved indgåelse af aftalen.  Nøglen kaldes nedenstående af KMD  Ocp-Apim-Subscription-Key |
| configuration.testData[?].employmentId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder KMD I2. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter. |
| configuration.testData[?].ssn | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder KMD I2. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter. |
| configuration.testData[?].email | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder KMD I2. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter. |
| configuration.testData[?].institutionDtrId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder KMD I2. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter. |
| configuration.testData[?].roles[?] | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  Disse data kan udelades hvis man har slået integrationen til, og faktisk kalder KMD I2. Hvis man har slået integrationen fra, og gerne vil returnere statiske data, kan man udfylde disse felter. |

## Test af korrekt opsætning

Når containeren er startet, så kan man teste output fra den vha curl eller lignende REST værktøj (postman er også en mulighed)

Det antages at containeren er udstillet mod localhost på port 8997 i nedenstående.

$ curl –header "ApiKey: xxxx" <http://localhost:8997/api/employments>

Hvis det hele er opsat korrekt, så returneres et JSON array med de brugere som findes i KMD I2 (som har et DTR ID tilknyttet).

## Yderligere API’er

Der udstilles 4 API endpoints, som kan bruges til oprettelse, nedlæggelse og opdatering af data i KMD I2. Disse er

**GET /api/employments**

Denne returnerer alle brugere der findes i KMD I2 som har et DTR ID tilknyttet. Stukturen er et JSON array hvor hvert element har denne form

{

”ssn”: ”0102037001”,

”employmentId”: 1,

”institutionDtrId”: ”G123456”,

”email”: ”bruger@kommune.dk”,

”mobilePhone”: ”1020304050”,

”roles”: [

”Teacher”,

”...”

]

}

**POST /api/employments**

Dette endpoint anvendes til at oprette en bruger. Som payload sendes en JSON struktur identisk med ovenstående. Man skal udelade employmentId, da denne dannes af KMD I2.

**POST /api/employments/{id}**

Dette endpoint bruges til at opdatere en bruger. Som {id} angiver man employmentId værdien på den bruger man ønsker at opdatere, og som payoad sender man ovenstående JSON struktur.

**DELETE /api/employments/{id}**

Dette endpoint anvendes til at slette en bruger. Som {id} angiver man employmentId på den bruger man ønsker at slette.

# OS2sync opsætning

Løsningen kan integrere med OS2sync, som man kan deploye som beskrevet på OS2sync websitet. Man skal bruge version 2.5.1 eller nyere

<https://www.os2sync.dk>

## Stubbet udgave

Man kan også opsætte en stubbet udgave, der returnerer et statisk sæt af data, hvilket kan være praktisk til test- og udviklingsformål. Der er et doc

Man kan hente en eksempel docker-compose.yml fil her

<https://github.com/OS2services/AulaDagtilbudIntegration/tree/master/docker>

der viser hvordan man kan opsætte denne stub-container til brug med Docker Compose.

Nedenstående er en gennemgang af de konfigurationsmuligheder der er til containeren.

### Konfiguration

Der publiceres images til anvendense på Dockerhub her, hvor man altid kan finde den nyeste udgave

<https://hub.docker.com/orgs/os2services>

Internt kører containeren på port 5000, men man kan vælge at mappe dette til en anden port når/hvis den udstilles udenfor det netværk containeren deployes i.

I environment sektionen af docker-compose.yml filen har man mulighed for at opsætte følgende værdier

|  |  |
| --- | --- |
| **Nøgle** | **Beskrivelse** |
| configuration.apiKey | Containeren udstiller et REST API som kan kaldes af de andre containere i deployment miljøet. Her angives den nøgle som skal anvendes når der foretages kald mellem containerne.  Kald foretaget uden denne nøgle vil blive afvist af denne container. |
| configuration.testData.users[?].uuid | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”uuid” udfyldes med UUID på en bruger som er tilknyttet en enhed med et DTR ID. |
| configuration.testData.users[?].ssn | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”ssn” udfyldes med CPR nummeret på brugeren. |
| configuration.testData.users[?].userId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”userId” udfyldes med AD brugernavnet på brugeren. |
| configuration.testData.users[?].phone | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”phone” udfyldes med mobilnummeret på brugeren. |
| configuration.testData.users[?].email | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”email” udfyldes email adressen på brugeren. |
| configuration.testData.users[?].dtrId | Dette er et array (man kan gentage denne indstilling flere gange, og udfylde værdierner 0,1,2,3,osv i feltet [?].  ”dtrId” udfyldes med det DTR ID som brugeren skal være tilknyttet. |
| configuration.testData.users[?].manager | Sættes til ”true” hvis denne person er leder for enheden. |

## Test af korrekt opsætning

Når containeren er startet, så kan man teste output fra den vha curl eller lignende REST værktøj (postman er også en mulighed)

Man kan teste opsætningen af både den stubbede container og den rigtige OS2sync på samme måde.

Det antages at containeren er udstillet mod localhost på port 5000 i nedenstående.

$ curl –header "ApiKey: xxxx" <http://localhost:5000/api/DtrId>

Hvis det hele er opsat korrekt, så returneres et JSON array med de brugere som findes i FK Organisation (som har et DTR ID tilknyttet).

# Opsætning af integrationsflow

Integrationens forretningslogik er samlet i en selvstændig container. Denne container kommunikerer alene med de omliggende containere, baseret på den REST snitflade disse udstiller internt. Man kan på den måde udskifte disse containere enten med stubbede integrationer, eller konkrekte integrationer mod forskellige eksterne systemer efter behov.

## Docker Container

Man kan hente en eksempel docker-compose.yml fil her

<https://github.com/OS2services/AulaDagtilbudIntegration/tree/master/docker>

der viser hvordan man kan opsætte denne container til brug med Docker Compose.

Nedenstående er en gennemgang af de konfigurationsmuligheder der er til containeren.

### Konfiguration

Der publiceres images til anvendense på Dockerhub her, hvor man altid kan finde den nyeste udgave

<https://hub.docker.com/orgs/os2services>

Integrationen udstiller ikke selv nogen snitflader, men afvikler et job på det konfigurede tidspunkt. Dette job udfører den fulde opdatering af data.

I environment sektionen af docker-compose.yml filen har man mulighed for at opsætte følgende værdier

|  |  |
| --- | --- |
| **Nøgle** | **Beskrivelse** |
| configuration.cron | Udfyldes med et CRON udtryk for hvornår og hvor ofte integratinoen skal afvikles. Eksempelvis kan den udfyldes med  ”0 0 3 \* \* \*”  Hvis den ønskes afviklet kl 03:00 hver nat, alle ugens dage  Eller  ”0 0 3,12,18 \* \* MON-FRI”  Hvis man ønsker den afviklet kl 03:00, 12:00 og 18:00 på hverdage.  Der angives 6 værdier i CRON udtrykket, hvor værdierne, i rækkefølge, betyder   1. Sekund (”0/15” betyder hvert 15. sekund, ”20” betyder på sekund-tallet 20, ”10,20,30” betyder på sekund-tallene 10, 20 og 30, og endeligt ”\*” for hvert sekund 2. Minut (samme syntax som for sekunder) 3. Time (samme syntax som for sekunder) 4. Dag (dag i måneden, ellers samme syntax som for sekunder) 5. Måned (hvilke måneder jobbet skal køre, ”\*” er alle måneder, og typisk det man ønsker) 6. Dage (”\*” for alle dage, men ”MON-FRI” for hverdage, og tilsvarende... der anvendes engelske forkortelser for ugedagene) |
| configuration.apiKey | Udfyldes med den ApiKey der ønskes anvendt ved manuelt kald af jobbet |
| configuration.roles.supportedRoles | Udfyldes med en liste af roller som skal forsøges udlæst fra rolle-registeret (AD/OS2rollekatalog).  Eksempel herpå er  ”Teacher,Pedagogue” |
| configuration.roles.url | Her angives URL endpoint på rolle-register containeren |
| configuration.roles.apiKey | Her angives den apiKey der er opsat i rolle-register containeren |
| configuration.organisation.url | Her angives URL endpoint på organisations containeren |
| configuration.organisation.apiKey | Her angives apiKey på organisations containeren |
| configuration.organisation.readManagers | Sættes til ”true” hvis man ønsker at udlæse ledere fra STS Organisation. Disse vil blive indlæst som egentlige tilhørsforhold på lige fod med andre tilhørsforhold. |
| configuration.organisation.implicitManagerRole | Sættes til ”true” hvis man ønsker at udlæste ledere automatisk skal tildeles rollen InstitutionManager (har ingen effekt hvis man ikke udlæser ledere) |
| configuration.registration.url | Her angives URL på institutions containeren |
| configuration.registration.apiKey | Her angives apiKey på institutions containeren |
| configuration.registration.dtrFilter | Optionelt. Kan udfyldes med en komma-separaret liste af DTR ID’er som integrationen er autoritativ for. Der er altid et implicit filter baseret på de DTR ID’er som kan udlæses fra STS Organisation, men på denne måde kan man indskrænke listen yderligere.  Undlad at sætte en værdi hvis man ikke ønsker yderligere filtrering end den implicitte. |

## Test af korrekt opsætning

Der udstilles et API på integrationen, der kan trigge det job der normalt afvikles skeduleret. Dette kan kaldes på følgende måde

$ curl –header "ApiKey: xxxx" <http://localhost:8755/api/trigger>

# Overvågning

Man bør overvåge applikationerne, hvilket kan gøres ved hjælp af 2 metoder. Det anbefales at man bruger begge metoder i samspil

## Overvågning af applikationens ”liveliness”

Alle docker containerne, på nær OS2sync, udstiller et API til at checke om applikationen er levende. Dette kaldes ved at tilgå dette endpoint

$ curl <http://localhost:xxxx/manage/health>

Bemærk at xxxx skal skiftes ud med port-nummeret. Ovenstående returnerer HTTP 200 hvis alt er ok. Hvis der er noget galt, så returneres en HTTP statuskode forskellig fra 200, og man kan i svaret se evt detaljer om fejlen. Man bør dog også tilgå logfilen for at se yderligere detaljer.

## Overvågning af applikations log

Alle applikationerne logger løbende informationer om hvad der sker i applikationen. Alle fejl som kræver menneskelig involvering logges på niveauet ERROR, og man kan derfor overvåge loggen, og blot scanne efter tekst-strengen ERROR. Hvis den optræder i loggen, så bør man reagere, og forholde sig til den konkrete fejl.