# Interfaces de la MiData

## Aperçu

Il existe en principe deux possibilités de tirer des données de la MiData:

### Jetons de service

Les comptes de service permettent de créer un compte personnel avec certains droits pour une application externe avec lequel on peut utiliser l'interface JSON. Les comptes de service sont créés par une personne autorisée par niveau et restent aussi utilisables si cette personne quitte le groupe ou est supprimée.

Spécification (en anglais)

### OAuth

La MiData est également fournisseuse de OAuth 2.0, c'est-à-dire qu'une application externe peut authentifier des utilisateurs via la MiData. Tu connais peut-être déjà ce principe si tu as déjà utilisé la connexion par Facebook ou par Google. L'application externe peut ensuite demander des informations sur l'utilisateur de la MiData si l'utilisateur l'autorise lui-même. La connexion à OAuth permet aussi à l'application externe d'utiliser les interfaces JSON au nom de l'utilisateur. L'application a donc les mêmes droits que l'utilisateur.

Spécification (en anglais)

Si tu souhaites utiliser la MiData en tant que fournisseur OAuth pour l'une de tes applications, tu dois remplir une demande OAuth. Le support IT du MSdS te fournira un accès au système productif dans la mesure où tous les prérequis sont satisfaits: Demande application OAuth

**Actualité:** Le 14 septembre 2021, les jetons d'utilisateurs personnels ont été marqués comme obsolètes. Ils ne seront donc plus disponibles pendant un certain temps. Les applications qui fonctionnaient jusqu'ici avec des jetons d'utilisateur personnels sont désormais accessibles par OAuth ou des jetons de service.

# Exemples de codes

## Jetons de service, exemple avec R

```r
# Install the required packages as follows:
# install.packages(c("jsonlite", "httr"))

library(jsonlite)
library(httr)

# Script parameters
baseurl <- "https://pbs.puzzle.ch"
token <- "d-Zx8kn-mKWaxXziYs62xVX2HdUdVKnSmLQYpQG-XznkbRD71g"
groupid <- 1

# Get MiData root group
rootElement <- fromJSON(paste(baseurl,"/groups/", groupid, ".json?token=", token, sep=""))
# Filter cantons
cantons = subset(rootElement$linked$groups, group_type == "Kantonalverband")
rownames(cantons) <- seq(length=nrow(cantons))

# Set a flag if the canton has an even id (just for fun)
cantons$even <- F
counter <- 1
for (id in cantons$id) {
       cantons$even[counter] <- ((as.numeric(cantons$id[counter]) %% 2) == 0)
       counter = counter + 1
}

# Print table of cantons
cantons
```

## OAuth – Ressources utiles

- Hitobito OAuth Flow
- Ruby (Doorkeeper, Devise, Omniauth) Example
- https://oauthdebugger.com
- https://www.oauth.com/playground/