Experiments with different integration approaches

Here we experiment with different midPoint-Grouper integration approaches. To be used internally by the project team. Please use **`laboratory`** branch as described below.

See Notes to Experimenters for lessons learned, speed bumps and glitches.

- Downloading and building the project
- Starting the project
- Description of data processing
    - Introduction
    - Source systems
    - midPoint and Grouper
        - Data in midPoint
        - Data in Grouper
    - Target systems
        - Target 1: Faculty portal
        - Target 2: Computer science students portal
        - Target 3: Generic mailing list application
        - Target 4: LDAP
- Design options
    - Discussion of the options

Downloading and building the project

This version of `demo/complex` is available on the `laboratory` branch. So you should use e.g. the following commands to download and build it:

```
$ cd ~
$ git clone -b laboratory https://github.internet2.edu/docker/midPoint_container.git

$ cd midPoint_container
$ ./build.sh
```

The project consists of several parts:

- `demo/complex` demonstrates an approach that makes midPoint responsible for all the interfacing with source and target systems, and Grouper responsible for maintaining the group membership. This is the same approach as was used in `demo/complex` from the beginning.
- `demo/complex2` is an alternative design that makes Grouper responsible for getting membership information from source systems.
- `demo/complex2s` is the same as `complex2` but midPoint-Grouper interface is simplified (hence "2s").

Because these compositions use the same ports, only one of them can be running at the same time.

Starting the project

Please visit the appropriate child page:

1. Design option 1: All interfacing via midPoint
2. Design option 2: SoR groups to Grouper
3. Design option 2s: SoR groups to Grouper, simplified

Description of data processing

In this section we describe the overall processing of the data. It is common for all design options. Differences are dealt with later.

Introduction

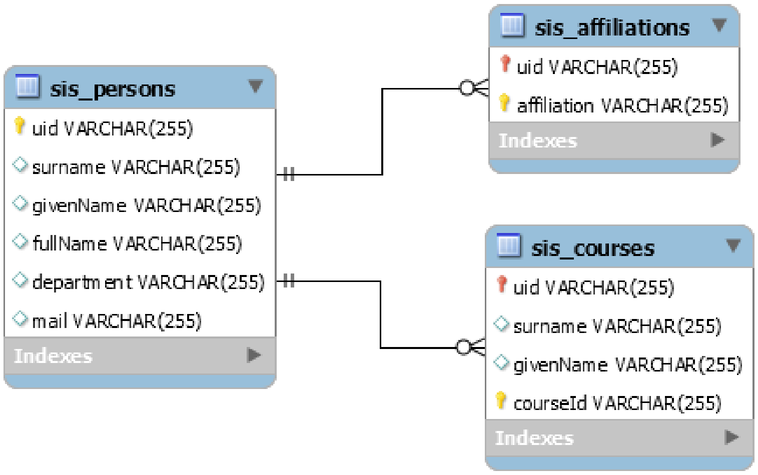
We need to:

1. fetch data from source systems (represented by a mock of a student information system),
2. get them into midPoint and Grouper where it is augmented and/or modified,
3. provision the data to target systems.

Let's have a look at these three areas.

Source systems

As a demonstration of source systems let's use the following (extremely simple) three tables:



Each person has:

1. zero or one department membership (e.g. `Business`, `Law`, and so on),
2. zero or more affiliations (`student`, `faculty`, `staff`, `member`, `alum`, `community`),
3. zero or more course enrollments (e.g. `CS251`, `MATH101`, and so on).

To summarize the data representation in SIS:

What	How	Example
person	row in `SIS_PERSONS` table	# uid, surname, givenName, fullName, department, mail 'bgasper', 'Gasper', 'Bill', 'Bill Gasper', 'Business', 'bgasper@example.edu'
person's department	`department` column	Business
person's affiliation	rows in `SIS_AFFILIATIONS` table	# uid, affiliation 'bgasper', 'alum'
person's courses	rows in `SIS_COURSES` table	# uid, surname, givenName, courseId 'bgasper', 'Gasper', 'Bill', 'CS251' 'bgasper', 'Gasper', 'Bill', 'MATH100'

(Actually, specific SQL representation is quite irrelevant, because SIS tables serve here only as a simplified version of a real academic information system.)

midPoint and Grouper

Via midPoint and Grouper we want do achieve the following:

1. To modify selected information from SoR by including and/or excluding given persons to/from given groups.
    a. This applies to affiliation information: For example we might want to state that although `bgasper` is listed under `alum` he should actually not be in `alum` but in `faculty`.
    b. Departmental and course information do not need to be changed in this way.
2. To create extra groups and manually maintain their members.
3. To create extra groups that aggregate information from other groups.

In these scenarios we decided to use group-management features of Grouper to implement the above requirements.

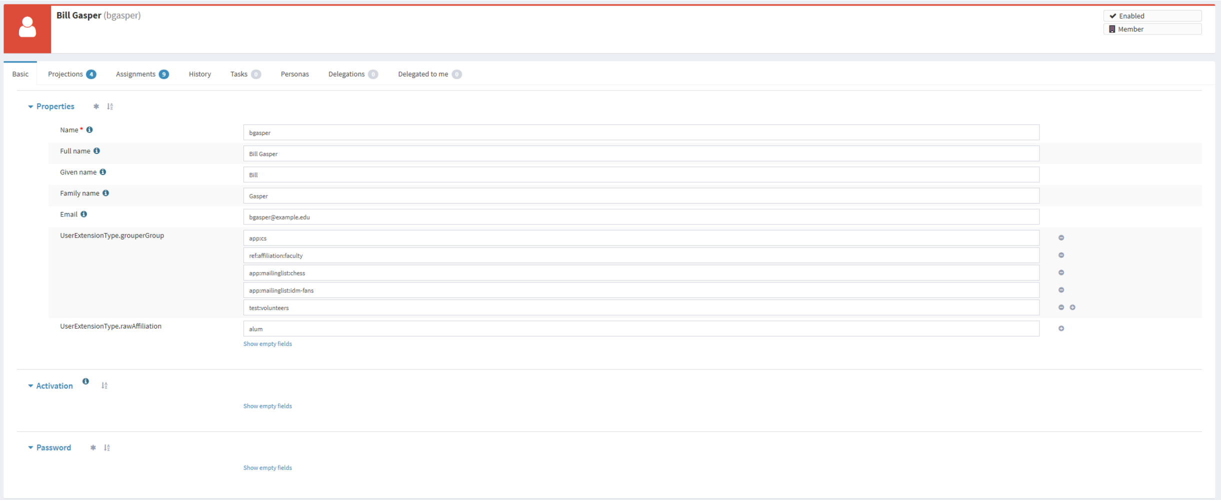
Data in midPoint

In midPoint the data is represented like this:

What	How	Example
person	user	bgasper
person's department	user org (of subtype `department`; under Departments org)	bgasper Business
person's refined affiliation	user org (of subtype `affiliation`; under Affiliations org)	bgasper Affiliation: faculty

person's courses	user org (of subtype `course`; under Courses org)	bgasper MATH100, CS251
person's mailing list membership	user org (of subtype `mailing-list`; under Mailing lists org)	bgasper Mailing list: chess, Mailing list: idm-fans
person's other membership	user org (of subtype `generic-group`; under Generic groups org)	bgasper test:volunteers, app:cs

An example:



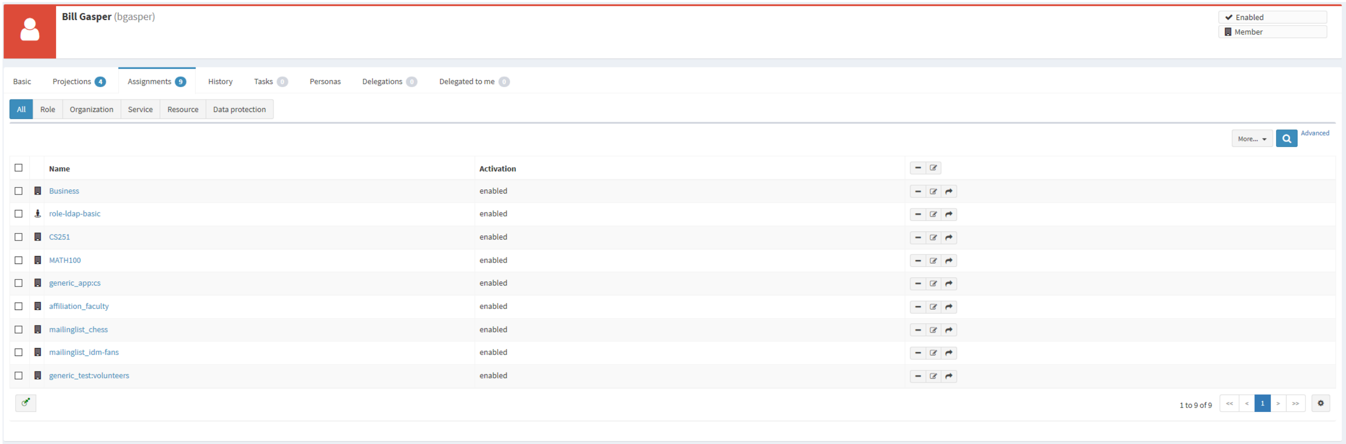
Bill Gasper (bgasper)

Properties

Name *	bgasper
Full name	Bill Gasper
Given name	Bill
Family name	Gasper
Email	bgasper@example.edu
UserExtensionType.grouperGroup	app:cs ref:affiliation:faculty app:mailinglist:chess app:mailinglist:idm-fans test:volunteers
UserExtensionType.rawAffiliation	alum

Activation

Password



Bill Gasper (bgasper)

	Name	Activation
□	Business	enabled
□	role-ldap-basic	enabled
□	CS251	enabled
□	MATH100	enabled
□	generic_app:cs	enabled
□	affiliation_faculty	enabled
□	mailinglist_chess	enabled
□	mailinglist_idm-fans	enabled
□	generic_test:volunteers	enabled

1 to 9 of 9

Relation targets (departments, affiliations, courses, mailing lists, other groups) are modeled as midPoint organizations.

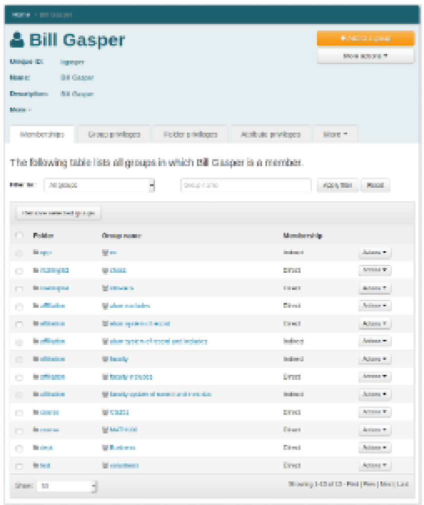
TODO some screenshots here

Data in Grouper

What	How	Example
person	subject referencing LDAP entry	uid=bgasper,ou=People,dc=internet2,dc=edu
person's department	membership in `ref:dept:XXX` group	ref:dept:Business
person's affiliation (from SoR)	membership in `ref:affiliation:XXX_systemOfRecord` groups	ref:affiliation:alum_systemOfRecord
person's affiliation (refined)	membership in `ref:affiliation:XXX` groups	ref:affiliation:faculty
person's courses	membership in `ref:course:XXX` groups	ref:course:MATH100, ref:course:CS251
mailing list membership	membership in `app:mailinglist:XXX` groups	app:mailinglist:chess, app:mailinglist:idm-fans
computer science course enrollment	membership in `app:cs` group	

any other membership	membership in respective groups	test:volunteers

An example:



Bill Gasper

Unique ID: bgasper

Name: Bill Gasper

Description: Bill Gasper

The following table lists all groups in which Bill Gasper is a member.

Folder	Group name	Membership
apps	[illegible]	Indirect
[illegible]	[illegible]	Direct
[illegible]	[illegible]	Direct
affiliation	[illegible]	Direct
affiliation	[illegible]	Direct
affiliation	[illegible]	Indirect
affiliation	faculty	Indirect
affiliation	faculty includes	Direct
affiliation	faculty system of record and includes	Indirect
courses	CS251	Direct
courses	MATH101	Direct
dept	Business	Direct
test	volunteers	Direct

Showing 1-13 of 13 · First | Prev | Next | Last

Target systems

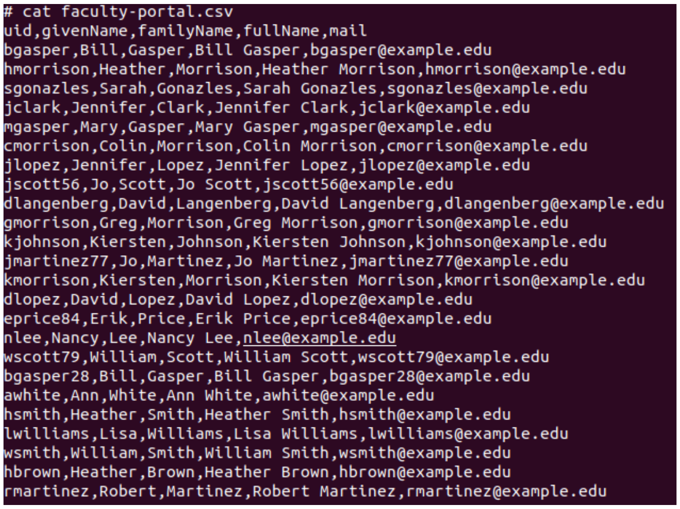
Target 1: Faculty portal

All users having affiliation of `faculty` (potentially modified in Grouper) should have a record in faculty portal database, carrying the following information: `uid`, `givenName`, `familyName`, `fullName`, `mail`.

⚠ This resource is temporarily created as a CSV.

Data representation:

What	How	Example
person's record	A database table row (temporary a line in CSV)	bgasper,Bill,Gasper,Bill Gasper,bgasper@example.edu (temporary in CSV)



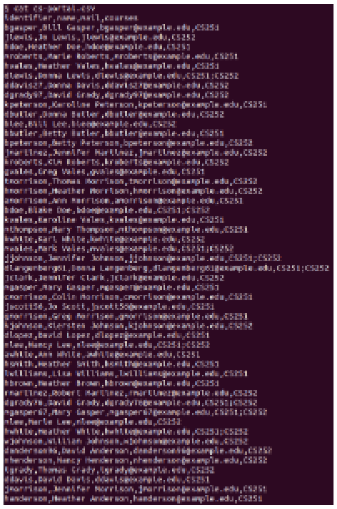
```
# cat faculty-portal.csv
uid,givenName,familyName,fullName,mail
bgasper,Bill,Gasper,Bill Gasper,bgasper@example.edu
hmorrison,Heather,Morrison,Heather Morrison,hmorrison@example.edu
sgonazles,Sarah,Gonazles,Sarah Gonazles,sgonazles@example.edu
jclark,Jennifer,Clark,Jennifer Clark,jclark@example.edu
mgasper,Mary,Gasper,Mary Gasper,mgasper@example.edu
cmorrison,Colin,Morrison,Colin Morrison,cmorrison@example.edu
jlopez,Jennifer,Lopez,Jennifer Lopez,jlopez@example.edu
jscott56,Jo,Scott,Jo Scott,jscott56@example.edu
dlangenberg,David,Langenberg,David Langenberg,dlangenberg@example.edu
gmorrison,Greg,Morrison,Greg Morrison,gmorrison@example.edu
kjohnson,Kiersten,Johnson,Kiersten Johnson,kjohnson@example.edu
jmartinez77,Jo,Martinez,Jo Martinez,jmartinez77@example.edu
kmorrison,Kiersten,Morrison,Kiersten Morrison,kmorrison@example.edu
dlopez,David,Lopez,David Lopez,dlopez@example.edu
eprice84,Erik,Price,Erik Price,eprice84@example.edu
nlee,Nancy,Lee,Nancy Lee,nlee@example.edu
wscott79,William,Scott,William Scott,wscott79@example.edu
bgasper28,Bill,Gasper,Bill Gasper,bgasper28@example.edu
awhite,Ann,White,Ann White,awhite@example.edu
hsmith,Heather,Smith,Heather Smith,hsmith@example.edu
lwilliams,Lisa,Williams,Lisa Williams,lwilliams@example.edu
wsmith,William,Smith,William Smith,wsmith@example.edu
hbrown,Heather,Brown,Heather Brown,hbrown@example.edu
rmartinez,Robert,Martinez,Robert Martinez,rmartinez@example.edu
```

Target 2: Computer science students portal

All computer science students (enrolled in CSxxx courses) should have a record in this system, providing the following information: `identifier` (i.e. `uid`), `name` (i.e. `fullName`), `mail`, computer science courses enrolled in.

Data representation:

What	How	Example
person's record	A line in CSV file	dlangenberg61,Donna Langenberg,dlangenberg61@example.edu,CS251;CS252



```
$ cat cs-portal.csv
identifier,name,mail,courses
bgasper,Bill Gasper,bgasper@example.edu,CS251
jlewis,Jo Lewis,jlewis@example.edu,CS252
hdoe,Heather Doe,hdoe@example.edu,CS251
mroberts,Marie Roberts,mroberts@example.edu,CS251
hvales,Heather Vales,hvales@example.edu,CS251
dlewis,Donna Lewis,dlewis@example.edu,CS251;CS252
ddavis27,Donna Davis,ddavis27@example.edu,CS252
dgrady97,David Grady,dgrady97@example.edu,CS252
kpeterson,Karoline Peterson,kpeterson@example.edu,CS251
dbutler,Donna Butler,dbutler@example.edu,CS252
blee,Bill Lee,blee@example.edu,CS252
bbutler,Betty Butler,bbutler@example.edu,CS251
bpeterson,Betty Peterson,bpeterson@example.edu,CS252
jmartinez,Jennifer Martinez,jmartinez@example.edu,CS252
kroberts,Kim Roberts,kroberts@example.edu,CS252
gvales,Greg Vales,gvales@example.edu,CS251
tmorrison,Thomas Morrison,tmorrison@example.edu,CS252
hmorrison,Heather Morrison,hmorrison@example.edu,CS252
amorrison,Ann Morrison,amorrison@example.edu,CS251
bdoe,Blake Doe,bdoe@example.edu,CS251;CS252
kvales,Karoline Vales,kvales@example.edu,CS251
mthompson,Mary Thompson,mthompson@example.edu,CS251
kwhite,Karl White,kwhite@example.edu,CS252
mvales,Mark Vales,mvales@example.edu,CS251;CS252
jjohnson,Jennifer Johnson,jjohnson@example.edu,CS251;CS252
dlangenberg61,Donna Langenberg,dlangenberg61@example.edu,CS251;CS252
jclark,Jennifer Clark,jclark@example.edu,CS252
mgasper,Mary Gasper,mgasper@example.edu,CS251
cmorrison,Colin Morrison,cmorrison@example.edu,CS251
jscott58,Jo Scott,jscott58@example.edu,CS251
gmorrison,Greg Morrison,gmorrison@example.edu,CS251
kjohnson,Kiersten Johnson,kjohnson@example.edu,CS252
dlopez,David Lopez,dlopez@example.edu,CS251
nlee,Nancy Lee,nlee@example.edu,CS251;CS252
awhite,Ann White,awhite@example.edu,CS251
hsmith,Heather Smith,hsmith@example.edu,CS251
lwilliams,Lisa Williams,lwilliams@example.edu,CS251
hbrown,Heather Brown,hbrown@example.edu,CS251
rmartinez,Robert Martinez,rmartinez@example.edu,CS252
dgrady78,David Grady,dgrady78@example.edu,CS251;CS252
mgasper67,Mary Gasper,mgasper67@example.edu,CS251;CS252
mlee,Marie Lee,mlee@example.edu,CS252
hwhite,Heather White,hwhite@example.edu,CS251;CS252
wjohnson,William Johnson,wjohnson@example.edu,CS252
danderson86,David Anderson,danderson86@example.edu,CS252
nhenderson,Nancy Henderson,nhenderson@example.edu,CS252
tgrady,Thomas Grady,tgrady@example.edu,CS252
ddavis,David Davis,ddavis@example.edu,CS251
jmorrison,Jennifer Morrison,jmorrison@example.edu,CS251
handerson,Heather Anderson,handerson@example.edu,CS251
```

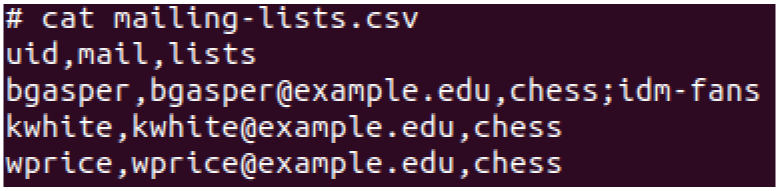
Target 3: Generic mailing list application

This application expects to get the set of pairs of (`listName`, `mail`) describing membership of individual mailing lists.

> ⚠ This resource is temporarily created as a CSV, represented as a set of (username,mail,list-of-mailing-lists) triples.

Data representation:

What	How	Example
mailing list membership	A line in CSV file (temporarily)	bgasper,bgasper@example.edu,chess;idm-fans



```
# cat mailing-lists.csv
uid,mail,lists
bgasper,bgasper@example.edu,chess;idm-fans
kwhite,kwhite@example.edu,chess
wprice,wprice@example.edu,chess
```

Of course, all this information required by targets 1-3 can be taken directly from LDAP. But we want here to simulate resources that need some extra processing (e.g. Box, Office365, and so on) leading to the use of a specific connector.

Target 4: LDAP

In order to provide information to a lot of other systems we need to maintain LDAP directory where each user has an `eduPerson` record with the following attributes or relations set (among others)

What	How	Example

person	`eduPerson` object with `givenName`, `sn`, `cn`, `mail` containing corresponding information from `sis_persons` table i.e. `givenName`, `surname`, `fullName`, `mail`, respectively	uid=bgasper,ou=People,
		dc=internet2,dc=edu

person's department	`businessCategory` attribute	Business

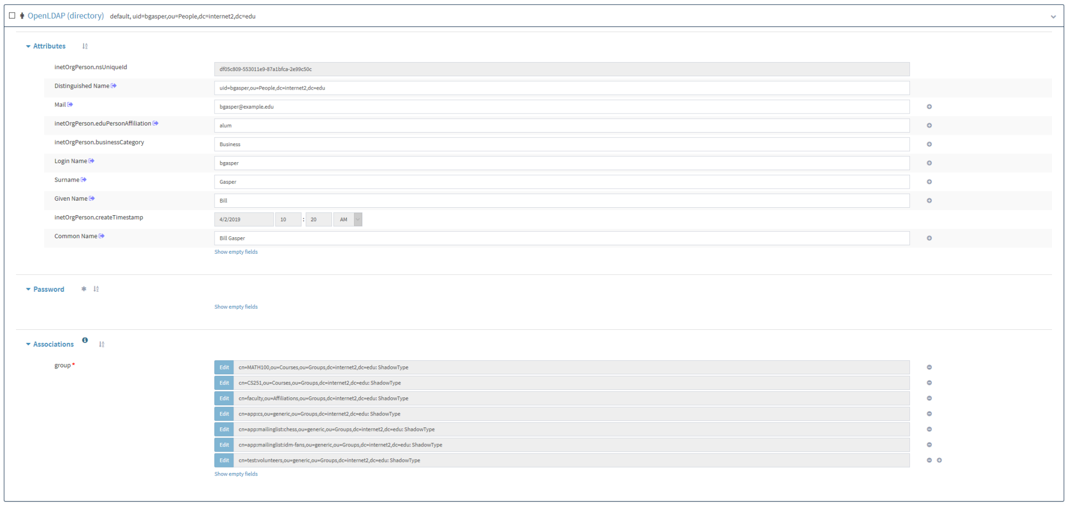
person's affiliation (refined by inclusion /exclusion)	group membership (in `ou=Affiliations,ou=Groups,dc=internet2,dc=edu` groups)	cn=faculty,ou=Affiliations,
		ou=Groups,dc=internet2,dc=edu

person's courses	group membership (in `ou=Courses,ou=Groups,dc=internet2,dc=edu` groups)	cn=MATH100,ou=Courses,ou=Groups,dc=internet2,dc=edu
		cn=CS251,ou=Courses,ou=Groups,dc=internet2,dc=edu

person's other Grouper groups	group membership (in `ou=Generic,ou=Groups,dc=internet2,dc=edu` groups)	cn=app:cs,ou=Generic,ou=Groups,dc=internet2,dc=edu
		cn=app:mailinglist:chess,ou=Generic,ou=Groups,dc=internet2,dc=edu
		cn=app:mailinglist:idm-fans,ou=Generic,ou=Groups,dc=internet2,dc=edu
		cn=test:volunteers,ou=Generic,ou=Groups,dc=internet2,dc=edu

person's midPoint-managed groups	group membership (in `ou=midpoint,ou=Groups,dc=internet2,dc=edu` groups)	cn=sysadmingroup,ou=midpoint,ou=Groups,dc=internet2,dc=edu

An example:



□ OpenLDAP (directory) default, uid=bgasper,ou=People,dc=internet2,dc=edu

Attributes

inetOrgPerson.nsUniqueId	df05c809-553011e9-87a1bfca-2e99c50c
Distinguished Name	uid=bgasper,ou=People,dc=internet2,dc=edu
Mail	bgasper@example.edu
inetOrgPerson.eduPersonAffiliation	alum
inetOrgPerson.businessCategory	Business
Login Name	bgasper
Surname	Gasper
Given Name	Bill
inetOrgPerson.createTimestamp	4/2/2019 10 : 20 AM
Common Name	Bill Gasper

Show empty fields

Password

Show empty fields

Associations

group *

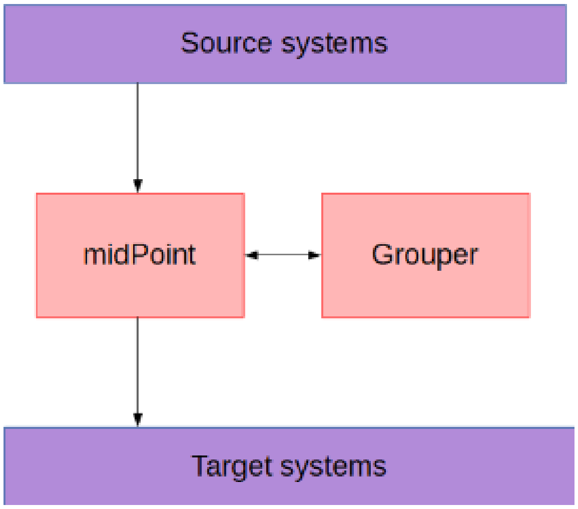
Edit cn=MATH100,ou=Courses,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=CS251,ou=Courses,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=faculty,ou=Affiliations,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=app:cs,ou=generic,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=app:mailinglist:chess,ou=generic,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=app:mailinglist:idm-fans,ou=generic,ou=Groups,dc=internet2,dc=edu: ShadowType
Edit cn=test:volunteers,ou=generic,ou=Groups,dc=internet2,dc=edu: ShadowType

Show empty fields

Design options

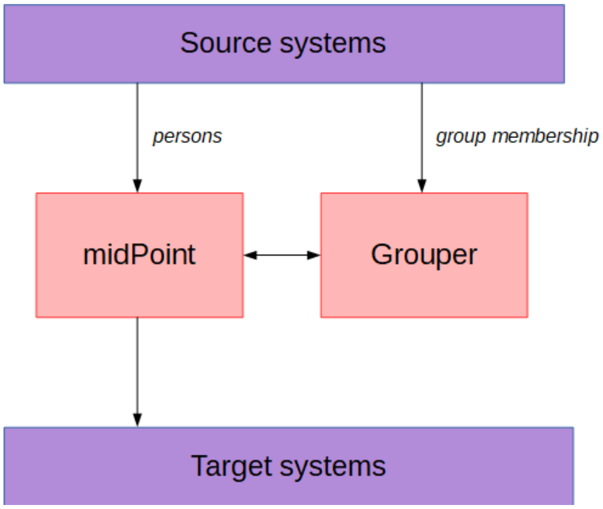
The two options differ in how group membership is transferred from source systems to Grouper.

In the first one ("All interfacing via midPoint") midPoint is solely responsible for getting the data and providing it in the cleaned form to Grouper:



The idea behind this option is to concentrate all interfacing at a single place: into midPoint, which has strong features in this area.

In the second one ("SoR groups to Grouper") midPoint gets only data about persons. Data about groups are transferred from source systems directly to Grouper:



The idea is to reduce the amount of data going through midPoint: if we ultimately want to have all groups in Grouper, and we only need some of them to be provisioned to target systems, it is not necessary to pull this information through midPoint.

The discussion on these options is at the end of this document (TODO); we might note, however, that both have their own rationale, and the ultimate selection among them depends on particular circumstances.

Discussion of the options

TODO

Option 1 requires the representation of "raw" SoR data that is flowing through midPoint via LDAP to Grouper. According to the current requirements this is the case of person's affiliation that is refined in Grouper.

So in option 1 we have the following additional data items:

What	Where	How	Example
person's affiliation (from SoR)	midPoint	user's `rawAffiliation` extension property	alum
	LDAP	`eduPersonAffiliation` attribute	alum

TODO

Note: Although Option 1 resembles the `demo/complex` on the `master` branch, it is a bit different. For example, raw affiliation (taken from SoR) is not represented as midPoint role membership but only as `rawAffiliation` extension attribute. The membership in generic groups taken from Grouper is represented by midPoint org membership and LDAP group membership, not just by extension property value setting as was in original `demo/complex` scenario.