CSV Connector - how to synchronize identities

Do you have an HR system that exports data into CSV files? You can try our brand new CSVConnConnector and easily transfer data into CzechIdM! This connector was developed for easy synchronization of CSV files into our system. The connector is mostly used for synchronization. Provisioning methods (update, create, delete) are implemented, too, but are not advisable for now.

In this tutorial, we will show you how to populate CzechIdm with identities from a CSV file given below as an example:

The CSV Connector is only meant for reconciliation!

Example file

If you want to follow this tutorial step by step, please copy these lines to a text editor and save the content as a CSV file (example.csv).

```
uid;username;firstname;lastname;desc
1;karamel;mel;kara;clovek 1
2;velbloud;bloud;vel;clovek 2
3;sakajavi;javi;saka;clovek 3
4;rucnik;nik;ruc;clovek 4
5;apache;che;apa;clovek 5
6;lalaalala;;lala;clovek 6
7;nugatek;tek;nuga;clovek 7
8;marenka;renka;ma;clovek 8
9;hodbod;bod;hod;clovek 9
10;blabol;bol;bla;clovek 10
11;karambol;bol;kara;clovek 11
```

As you can see, the CSV file contains a header.

Dependency to CzechIdm

First of all, you need to enable the CSV connector in CzechIdM

Import .jar file into CzechIdM library

1) Please, download this LINK to your computer.

2) Then copy it to YourServer(tomcat)/webapps/idm/WEB-INF/lib/

3) You also need to modify /webapps/idm/WEB-INF/lib/idm-ic-9.0.0-SNAPSHOT.jar and edit module-ic.properties here. It should look like this.

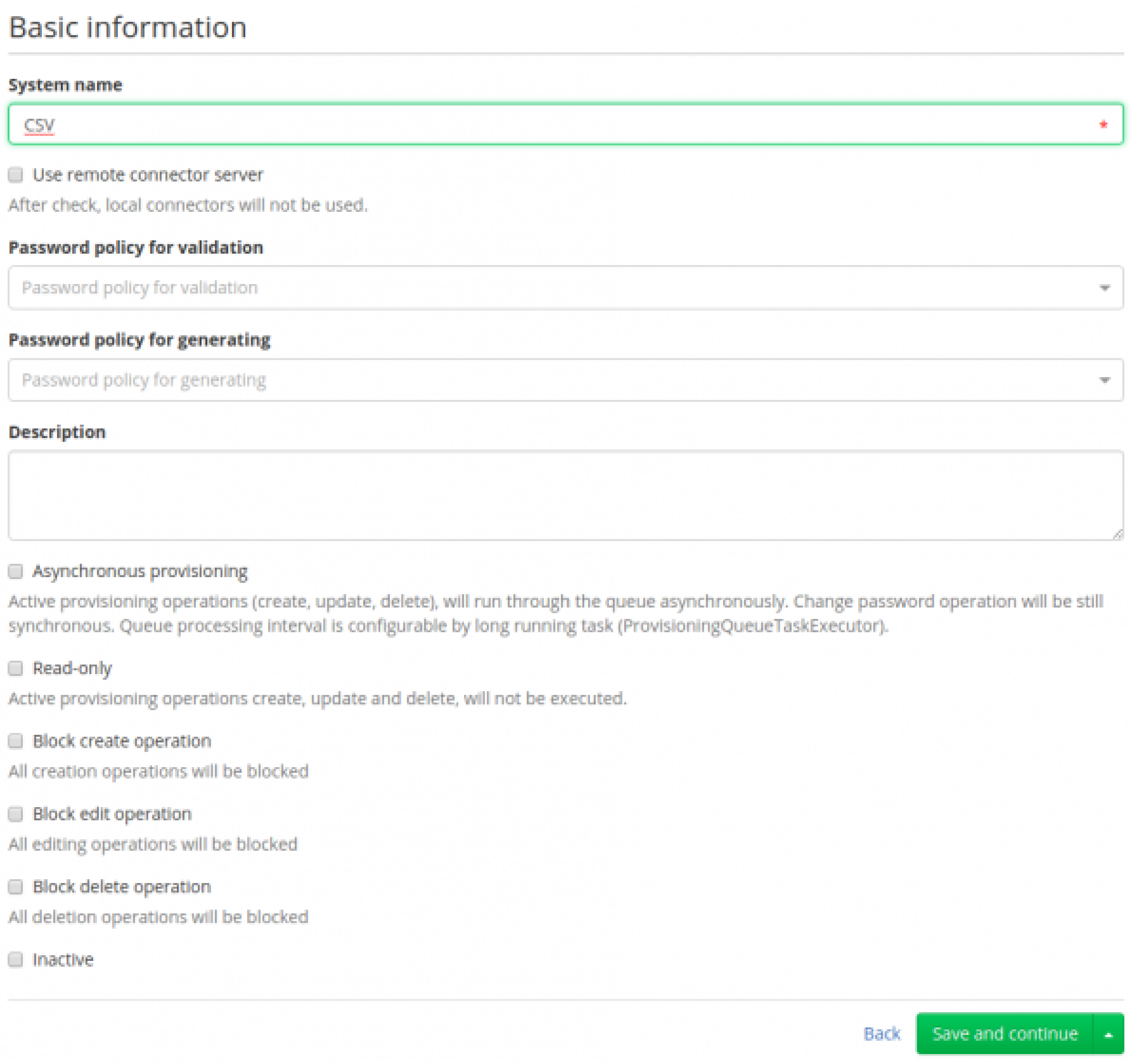
Versions may vary!

```
ic.localconnector.packages=net.tirasa.connid,eu.bcvsolutions.idm.vs.connecto
r,eu.bcvsolutions.idm.connectors.csv
```

4) You are now ready to do the next steps.

Creating a new system

Now you go back to CzechIdM. First of all, go to *Systems* and here *Add* a new system.


Basic information

System name: CSV

☐ Use remote connector server
After check, local connectors will not be used.

Password policy for validation: Password policy for validation

Password policy for generating: Password policy for generating

Description:

☐ Asynchronous provisioning
Active provisioning operations (create, update, delete), will run through the queue asynchronously. Change password operation will be still synchronous. Queue processing interval is configurable by long running task (ProvisioningQueueTaskExecutor).

☐ Read-only
Active provisioning operations create, update and delete, will not be executed.

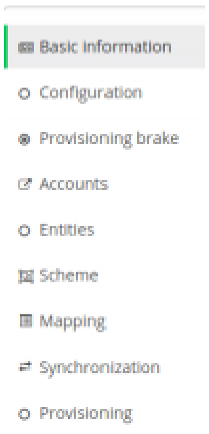
☐ Block create operation
All creation operations will be blocked

☐ Block edit operation
All editing operations will be blocked

☐ Block delete operation
All deletion operations will be blocked

☐ Inactive

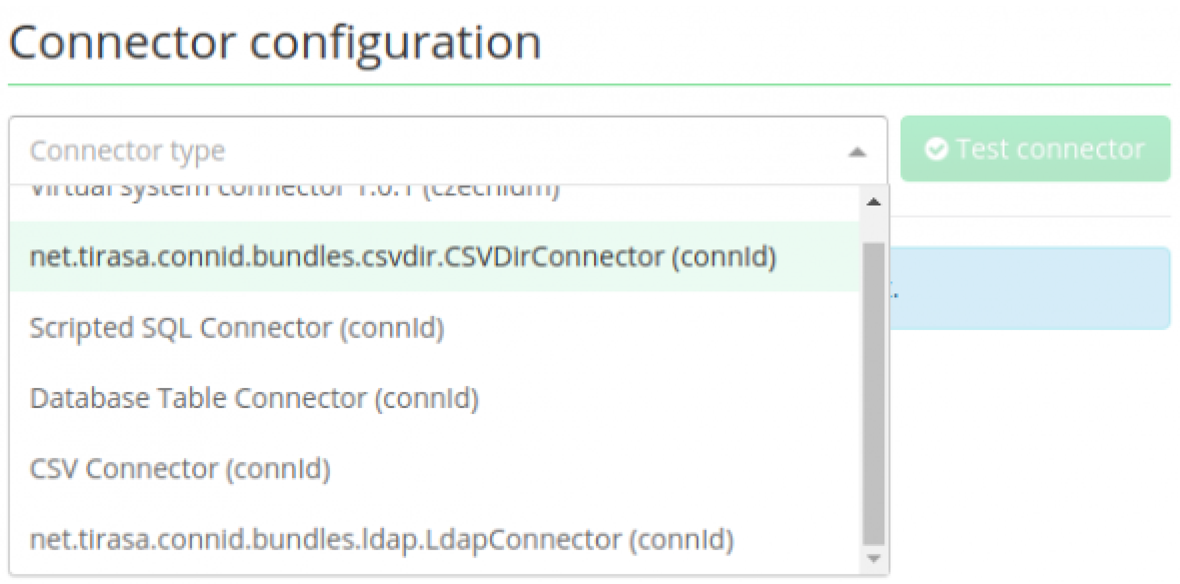
Fill in the name. The other options such as **Password policies**, **Description** or **Checkboxes below** are optional. You can just skip them or read more on the subject in generic system connection.



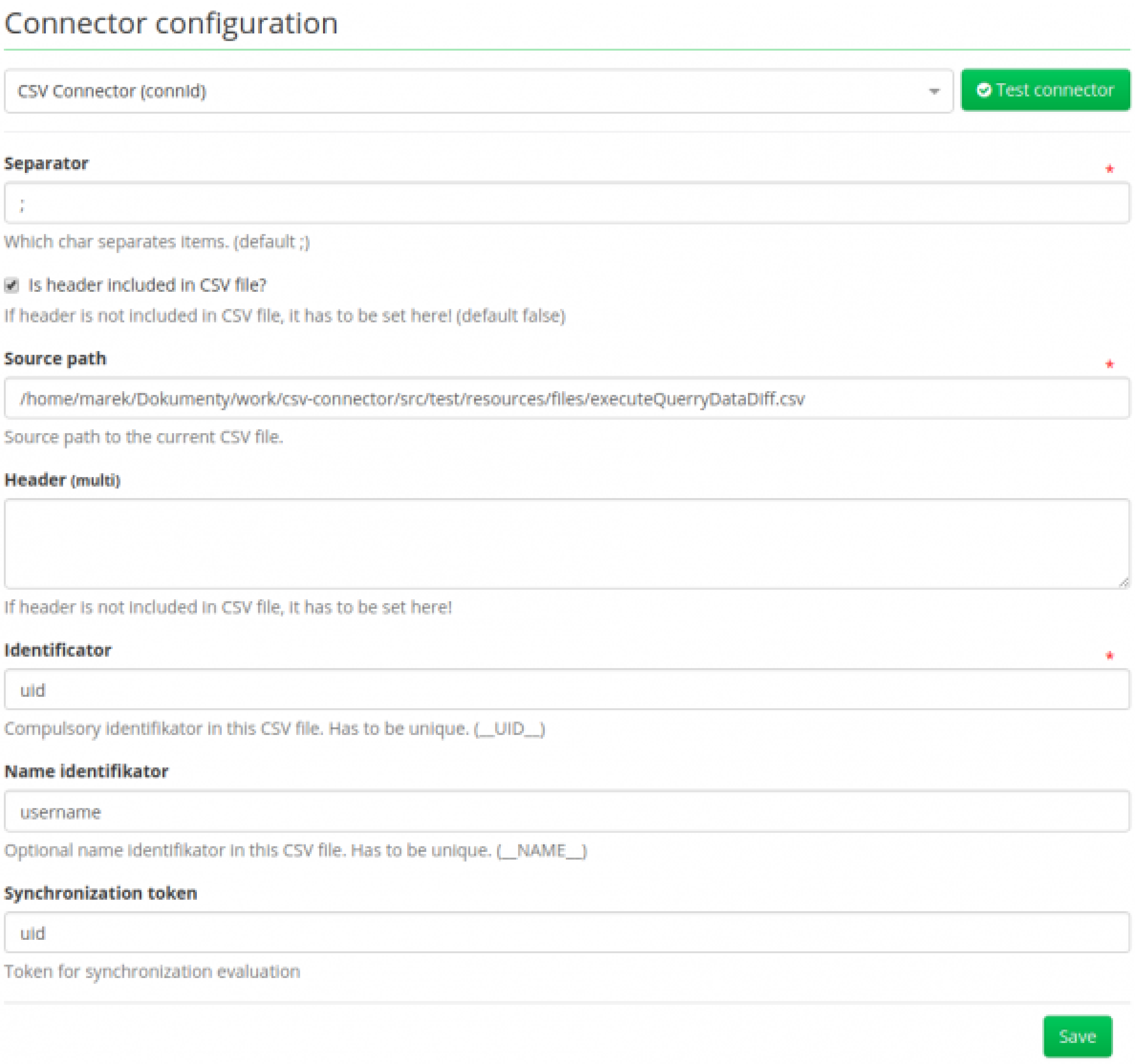
After saving the previous step you will see the (left-hand side) menu.

Configuration

Now we can reach **Configuration** page, where we need to set all properties for our CSV file. Look at the picture below:


Connector configuration
Connector type
Test connector
virtual system connector 1.0.1 (czechidm)
net.tirasa.connid.bundles.csvdir.CSVDirConnector (connId)
Scripted SQL Connector (connId)
Database Table Connector (connId)
CSV Connector (connId)
net.tirasa.connid.bundles.ldap.LdapConnector (connId)

First, we need to set our CSV Connector (connId), then all other needed information will pop up. Follow the instructions for each property.


Connector configuration

CSV Connector (connId)

Test connector

Separator *

;

Which char separates items. (default ;)

☑ Is header included in CSV file?

If header is not included in CSV file, it has to be set here! (default false)

Source path *

/home/marek/Dokumenty/work/csv-connector/src/test/resources/files/executeQuerryDataDiff.csv

Source path to the current CSV file.

Header (multi)

If header is not included in CSV file, it has to be set here!

Identificator *

uid

Compulsory identifikator in this CSV file. Has to be unique. (__UID__)

Name identifikator

username

Optional name identifikator in this CSV file. Has to be unique. (__NAME__)

Synchronization token

uid

Token for synchronization evaluation

Save

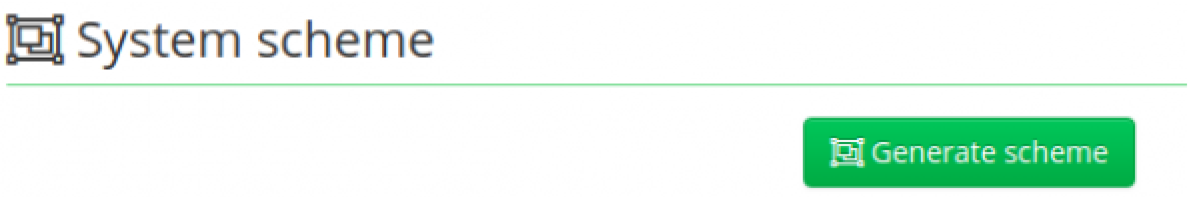
- **Separator** - character between two different columns
- **Source path** - path to the CSV file, which needs to be imported
- **Checkbox included** - if the first line of CSV file includes the header
- **Header** - if the file doesn't include the header, we need to write the header here
- **Identifier** - Which column holds the identifier
- **Name identifier** - This column is not mandatory for synchronization, you will need it for provisioning as secondary id.
- **Synchronization token** - token for repeated synchronization (not reconciliation) - usually DateTime

After we filled all information needed, just click **save** and **Test connector** and see if everything went all right. If some exception pops up, follow the instruction given. If you are following my example just fill every single item the same as shown in the picture.

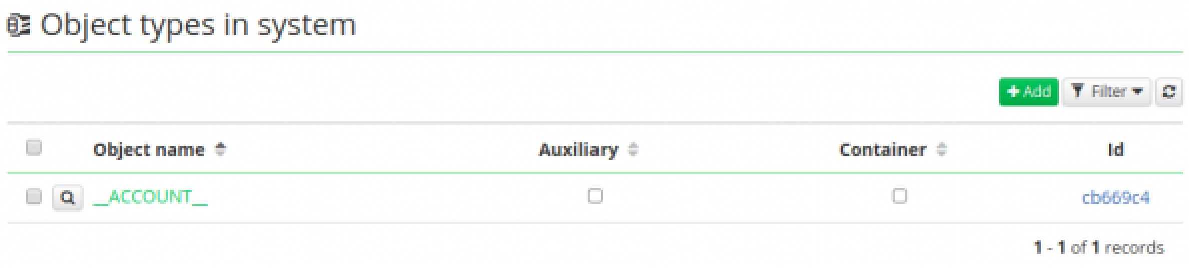
Scheme

When the connection works, we need to specify what attributes (CSV columns) will be synchronized. We call this a **Scheme**. Just click on the button to generate a scheme and then a new schema should appear at the bottom of the page.

1) Click on *Generate schema*



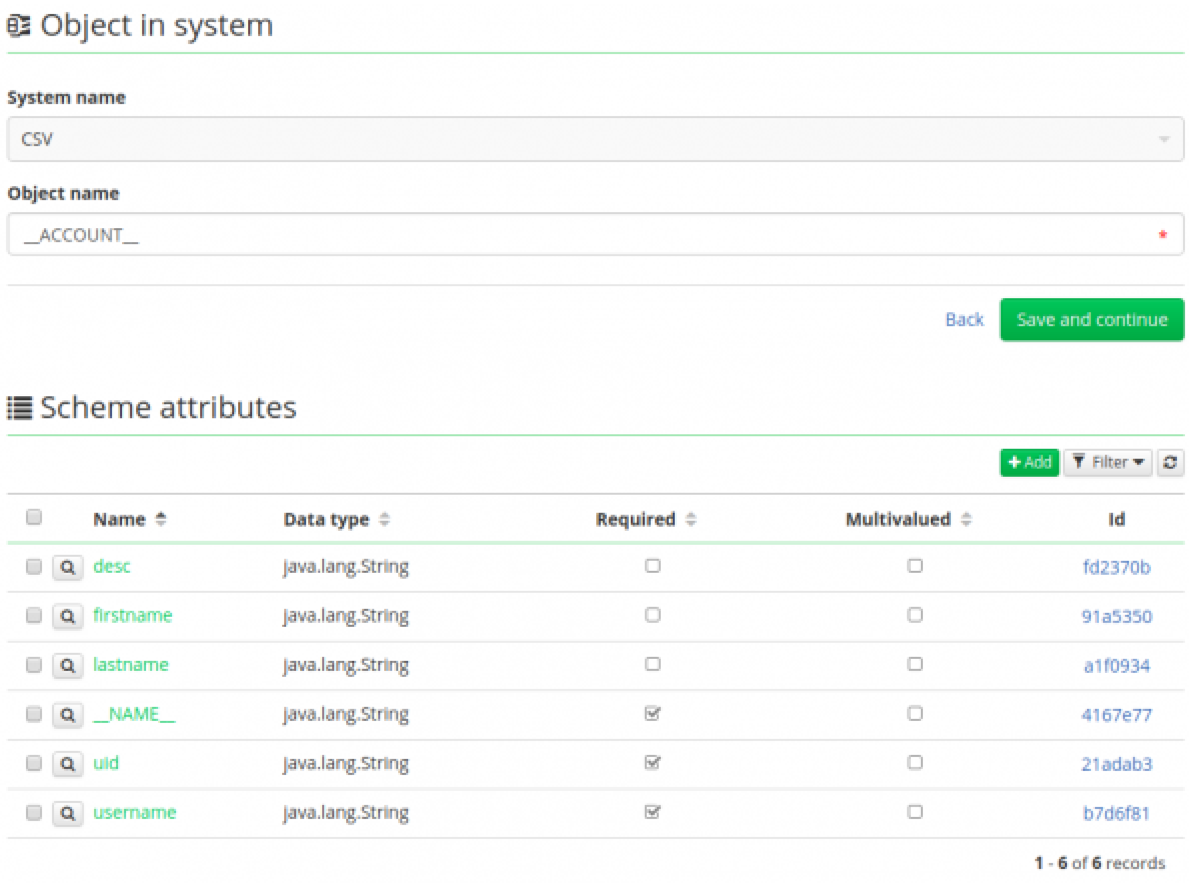
2) The schema should be created below for the respective object (in our example, it is ACCOUNT - the basic type of synchronized objects)


Object types in system

	Object name	Auxiliary	Container	Id
	__ACCOUNT__	☐	☐	cb669c4

1 - 1 of 1 records

3) Click on the magnifying glass and the following page will pop up


Object in system

System name: CSV

Object name: __ACCOUNT__

Scheme attributes

	Name	Data type	Required	Multivalued	Id
	desc	java.lang.String	☐	☐	fd2370b
	firstname	java.lang.String	☐	☐	91a5350
	lastname	java.lang.String	☐	☐	a1f0934
	__NAME__	java.lang.String	✓	☐	4167e77
	uid	java.lang.String	✓	☐	21adab3
	username	java.lang.String	✓	☐	b7d6f81

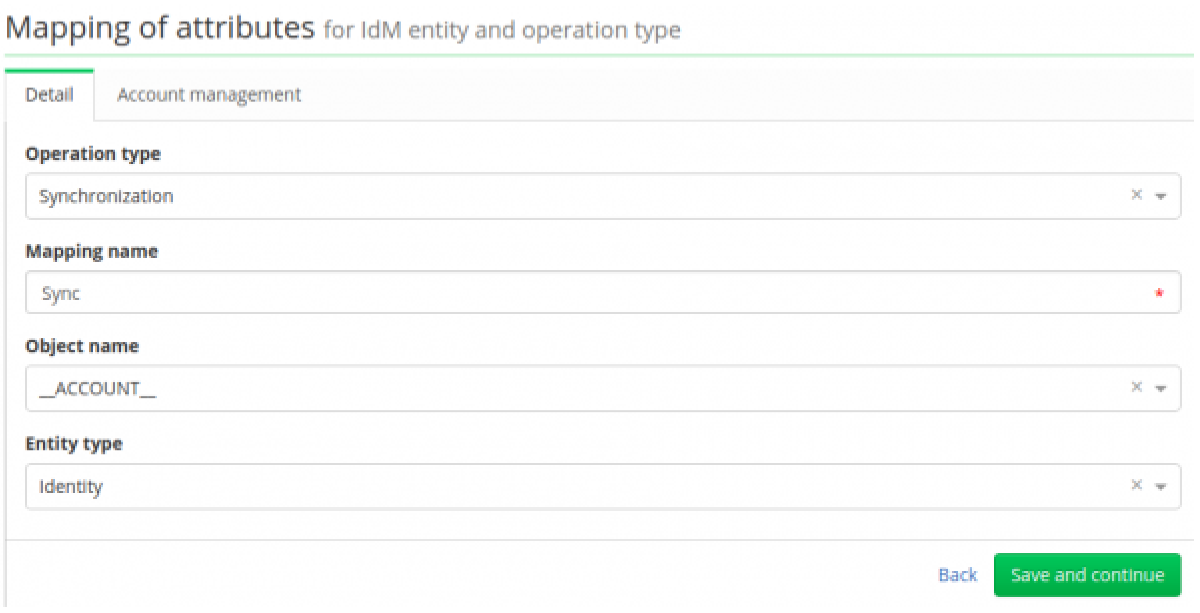
1 - 6 of 6 records

4) You can see that all items from our header were created as attributes here (desc, firstname, lastname…). You wonder what does the \NAME\ stand for? In the configuration window, there is the option **Name**, which is exactly what the \NAME\ is (again in our example, it is the username).

5) Please check if every single item from your header is present, only then proceed with the following steps in this tutorial.

Mapping

We have our schema created but now we need to map all items in the schema to actual attributes of Identity in CzechIdM.

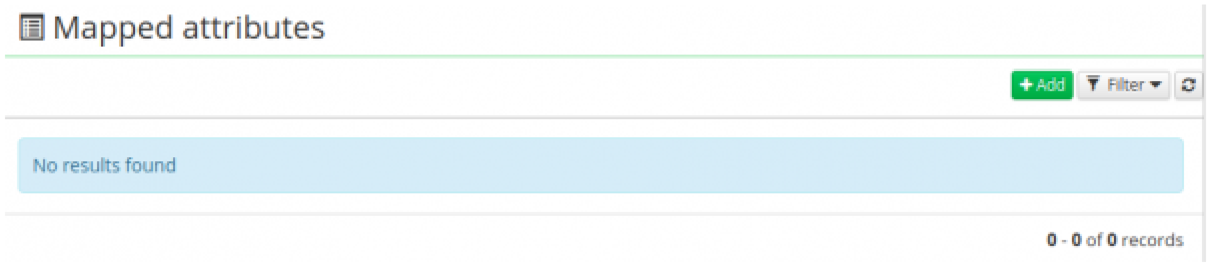

Mapping of attributes for IdM entity and operation type

Operation type	Synchronization
Mapping name	Sync
Object name	__ACCOUNT__
Entity type	Identity

1) Fill all needed attributes according to this picture if you follow my example.

- **Operation type** - type of operation. Synchronization for input data into CzechIdm and Provisioning for writing data into our CSV file (example.csv).
- **Mapping name** - the name which will be then used in synchronization/provisioning according to operation type.
- **Object name** - currently we have just account
- **Entity type** - Type of entity which will be created - can be Identity (our example), Tree, Role and so on. Look at the list given for other.

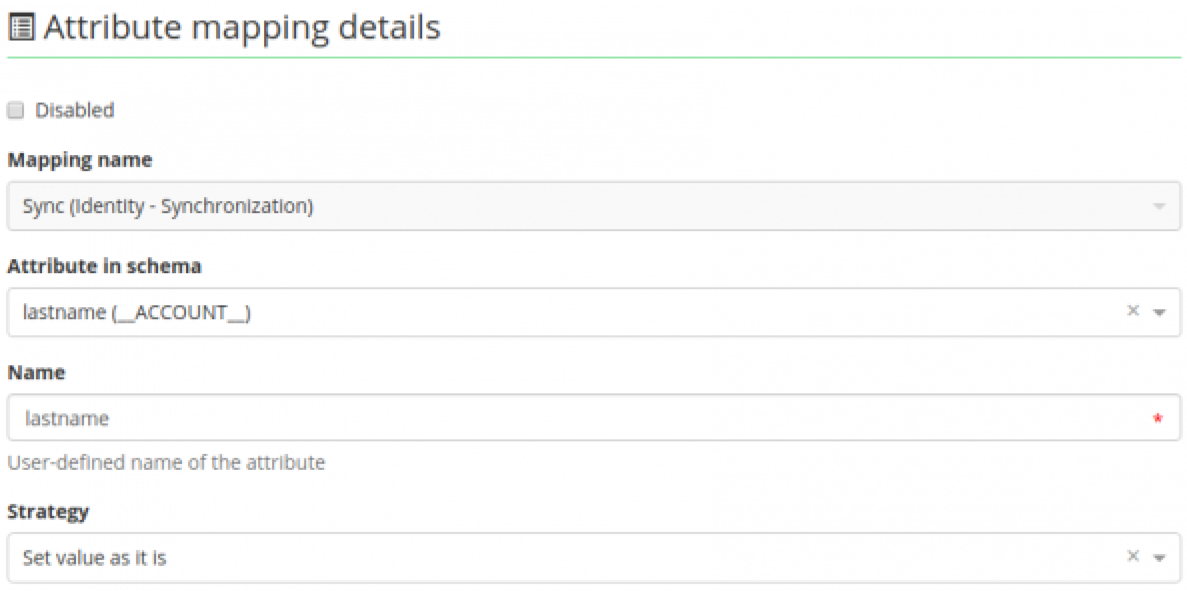
2) Click Save and continue.


Mapped attributes

No results found

0 - 0 of 0 records

3) Now the blank table is shown at the bottom of this page. There is option add so we click on that.


Attribute mapping details

☐ Disabled

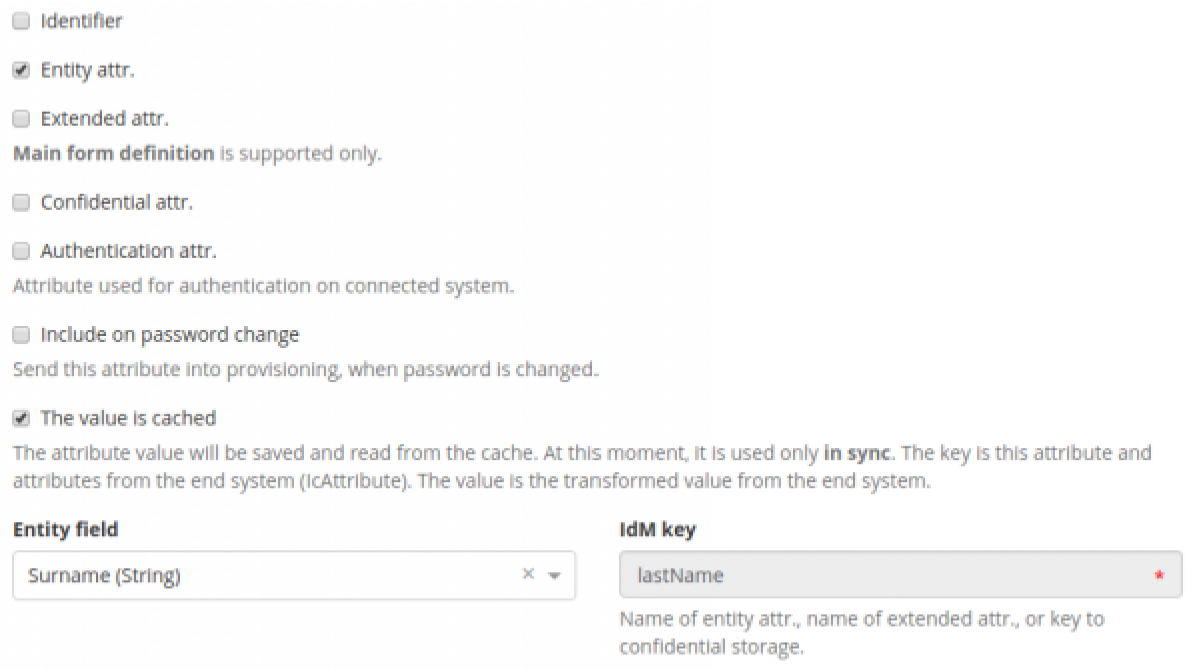
Mapping name
Sync (Identity - Synchronization)

Attribute in schema
lastname (__ACCOUNT__)

Name
lastname *
User-defined name of the attribute

Strategy
Set value as it is

4) Attribute mapping detail - this is the page where you chose the item from the schema and actually map it to the attribute of the object, we have chosen in the previous step. First, find **Attribute in schema** in the table of options for this select box. **Name** following will be set automatically. **Warning**: if we change our mind and set **Attribute in schema** to different option later, it won't change name automatically.


☐ Identifier

☑ Entity attr.

☐ Extended attr.
Main form definition is supported only.

☐ Confidential attr.

☐ Authentication attr.
Attribute used for authentication on connected system.

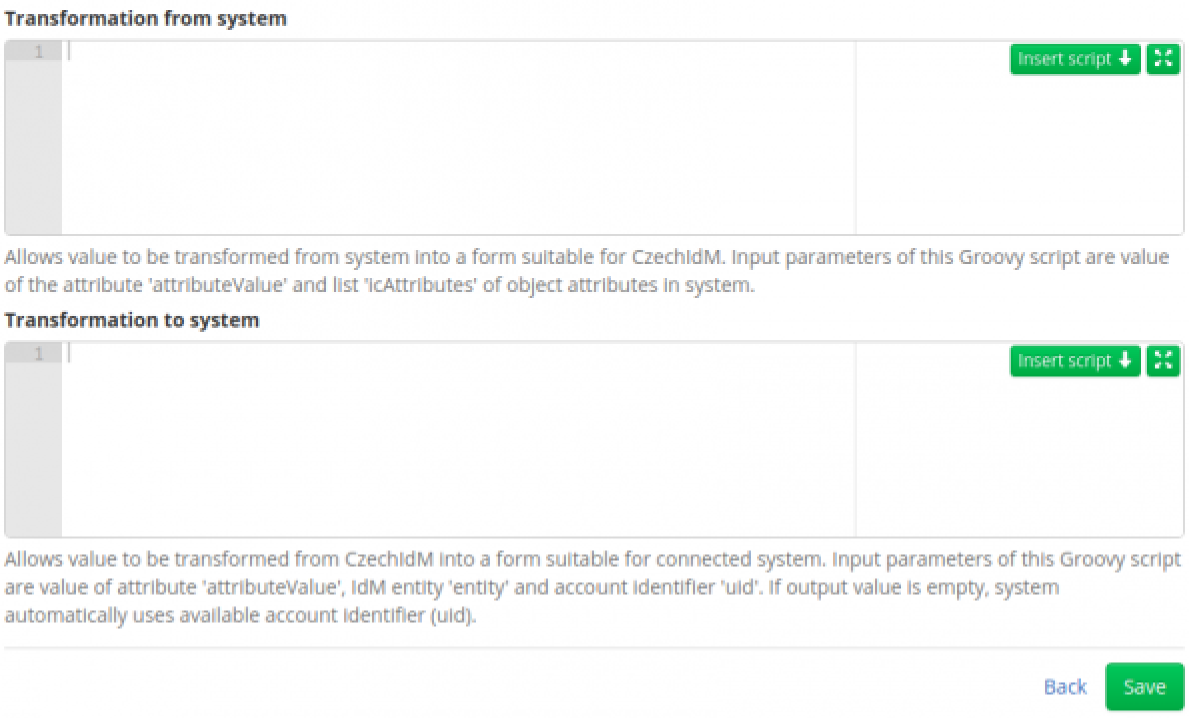
☐ Include on password change
Send this attribute into provisioning, when password is changed.

☑ The value is cached
The attribute value will be saved and read from the cache. At this moment, it is used only **in sync**. The key is this attribute and attributes from the end system (IcAttribute). The value is the transformed value from the end system.

Entity field
Surname (String)

IdM key
lastName *
Name of entity attr., name of extended attr., or key to confidential storage.

5) Attribute in CzechIdM and checkboxes with other info about attribute.

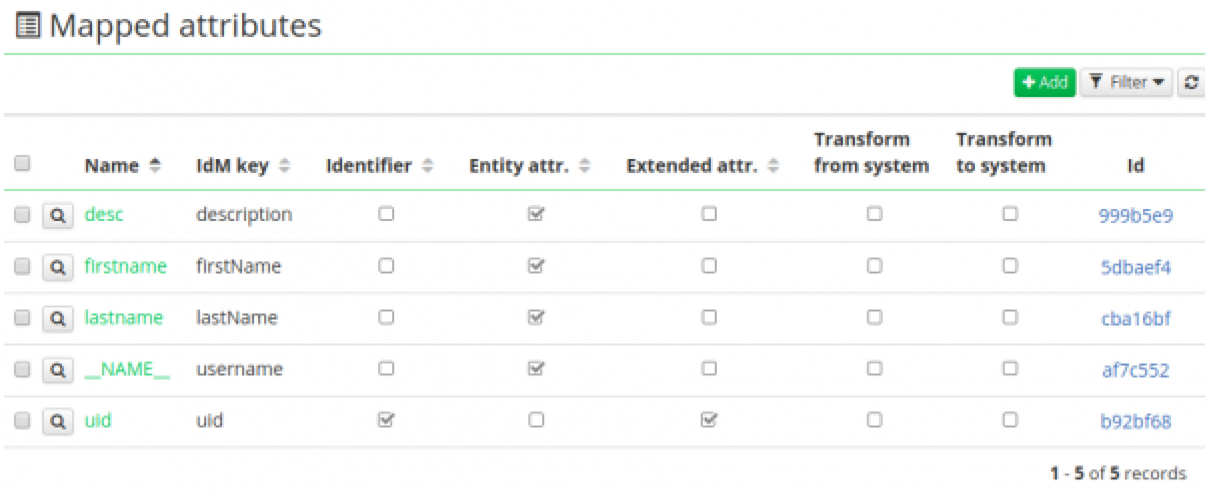

Transformation from system

Allows value to be transformed from system into a form suitable for CzechIdM. Input parameters of this Groovy script are value of the attribute 'attributeValue' and list 'icAttributes' of object attributes in system.

Transformation to system

Allows value to be transformed from CzechIdM into a form suitable for connected system. Input parameters of this Groovy script are value of attribute 'attributeValue', IdM entity 'entity' and account identifier 'uid'. If output value is empty, system automatically uses available account identifier (uid).

6) Fill all select boxes as shown in the pictures. We won't use transformations so according to the last picture we leave these two boxes blank.


Mapped attributes

	Name	IdM key	Identifier	Entity attr.	Extended attr.	Transform from system	Transform to system	Id
☐	desc	description	☐	☑	☐	☐	☐	999b5e9
☐	firstname	firstName	☐	☑	☐	☐	☐	5dbaef4
☐	lastname	lastName	☐	☑	☐	☐	☐	cba16bf
☐	__NAME__	username	☐	☑	☐	☐	☐	af7c552
☐	uid	uid	☑	☐	☑	☐	☐	b92bf68

1 - 5 of 5 records

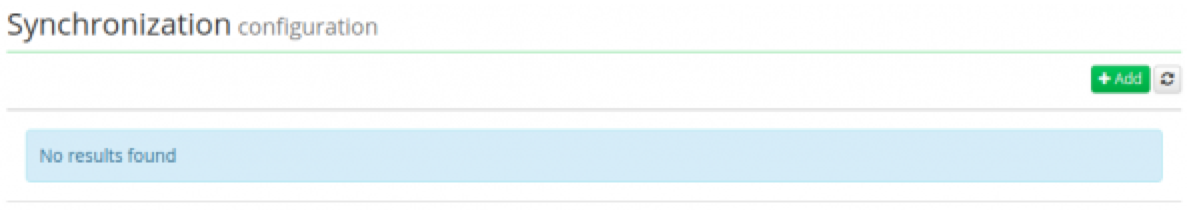
7) Make sure you mapped all attributes you need - the table should look like this.

For our example, we chose one of the items - **lastname**. All other items will be similar. Just for **uid** we need to choose select box *Identifier* and *Extended attribute*. Then just type to **IdM key** - uid.

If you want to know more about this topic, there is also the whole tutorial on mapping attributes in this LINK.

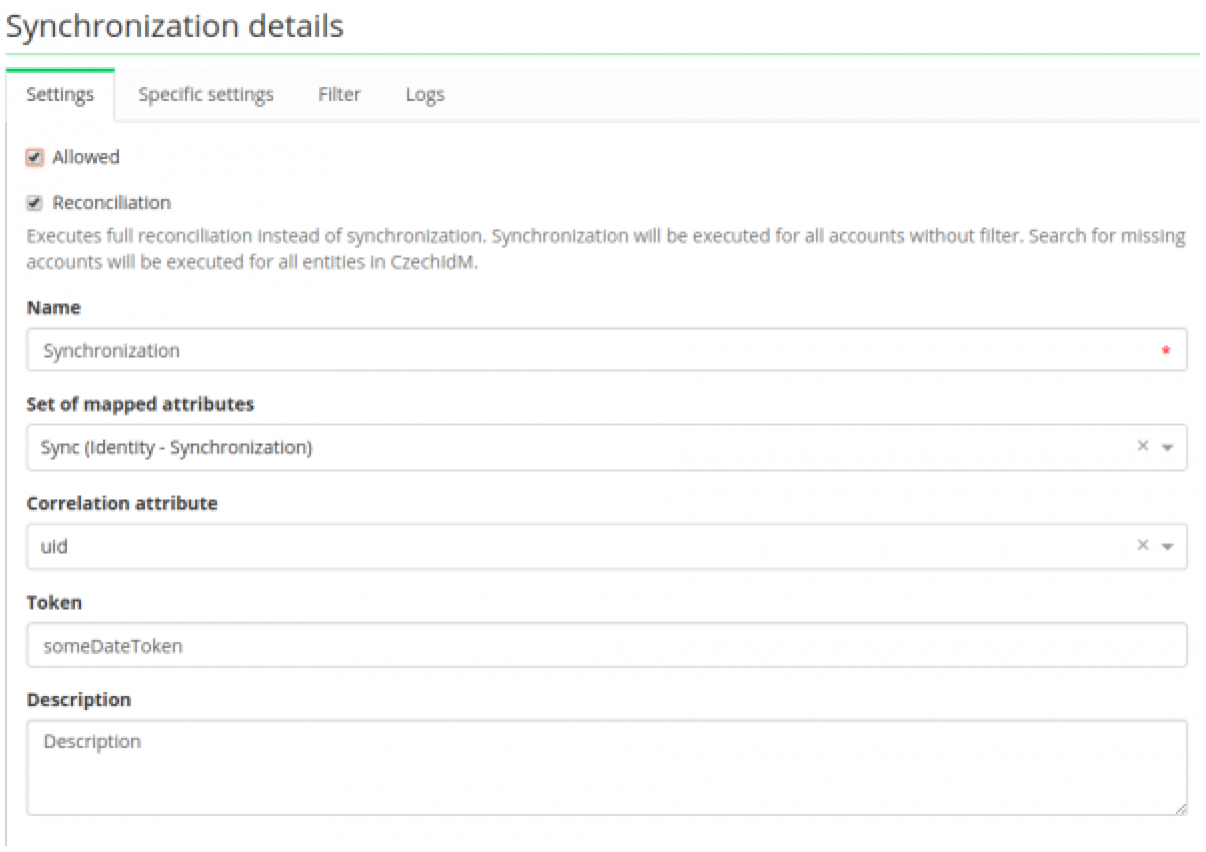
Synchronization

It starts with the addition of new synchronization.


Synchronization configuration

No results found

We do not use synchronization tokens (timestamps), so every time we will synchronize all data = reconciliation.


Synchronization details

☑ Allowed

☑ Reconciliation

Executes full reconciliation instead of synchronization. Synchronization will be executed for all accounts without filter. Search for missing accounts will be executed for all entities in CzechIdM.

Name: Synchronization *

Set of mapped attributes: Sync (Identity - Synchronization)

Correlation attribute: uid

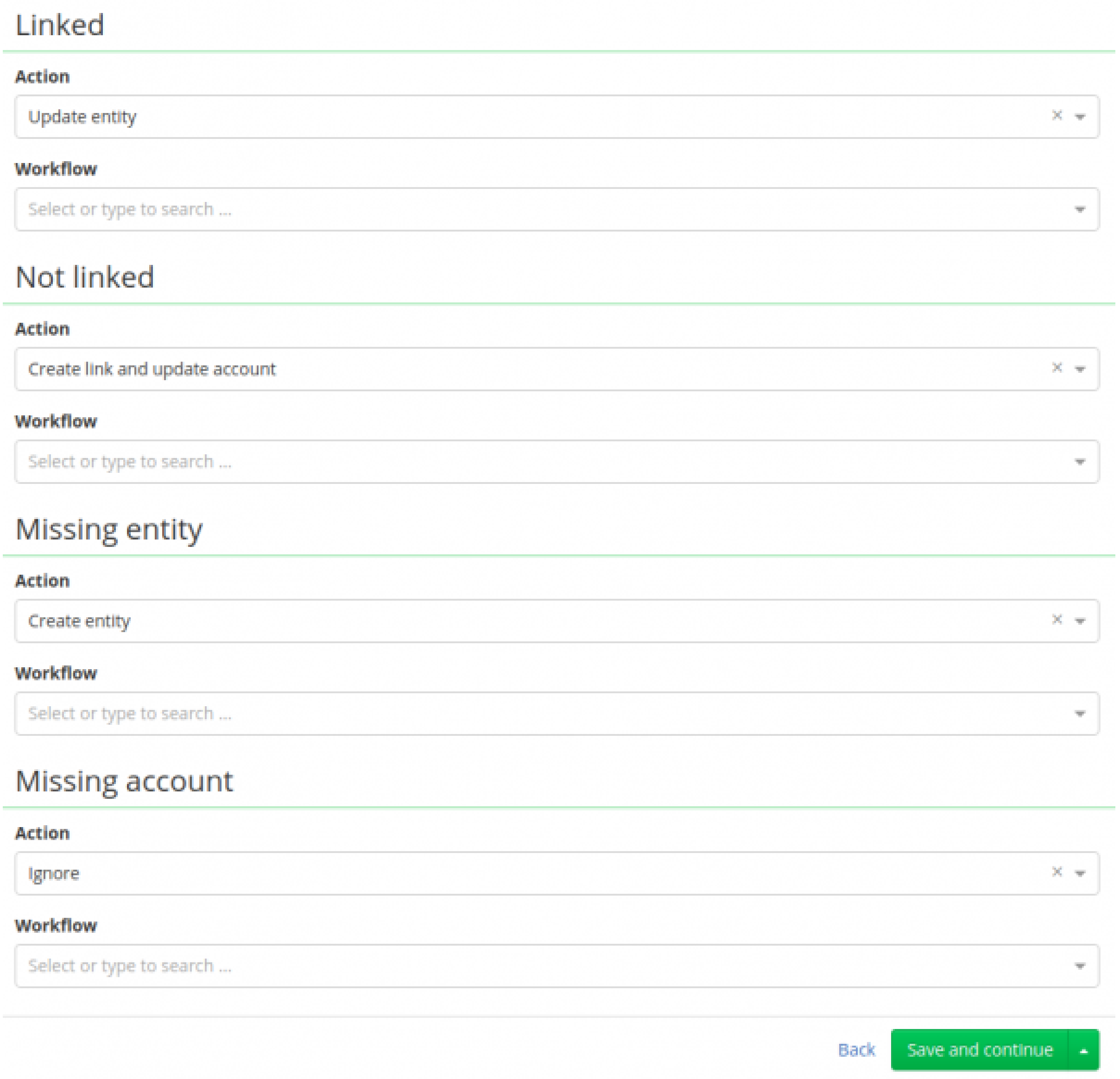
Token: someDateToken

Description: Description

1) We just simply have to set **Allowed** and **Reconciliation** - first, attribute only says that it can be run and second says that no sync. The token is given.

2) Choose your **Name** and **Set of mapped attributes** you created in previous steps.

3) **Correlation attribute** is the attribute which should connect both attribute in CzechIdM and item in the example.csv.



Linked

Action: Update entity

Workflow: Select or type to search ...

Not linked

Action: Create link and update account

Workflow: Select or type to search ...

Missing entity

Action: Create entity

Workflow: Select or type to search ...

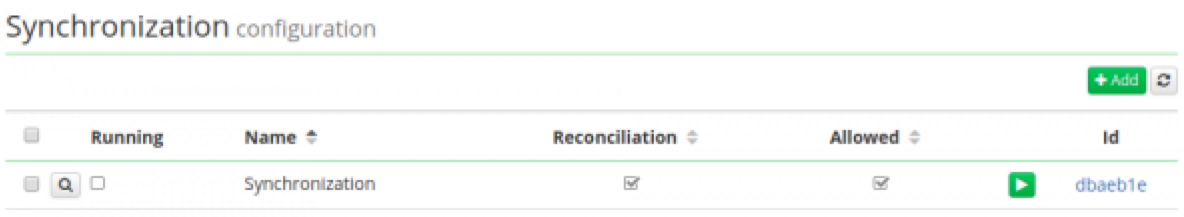
Missing account

Action: Ignore

Workflow: Select or type to search ...

4) There are more options but in this case, we won't change anything. For a brief explanation of what to set here follow this LINK.

5) After save and continue we can run our **Synchronization**.



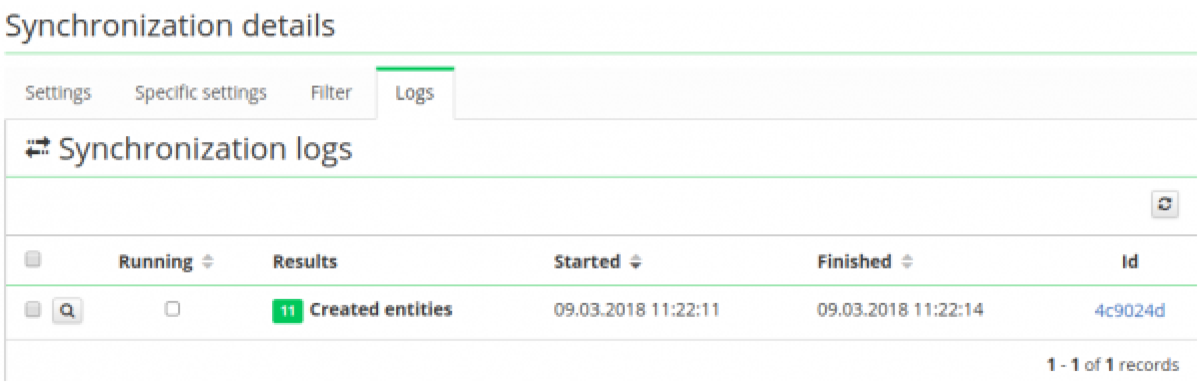
Synchronization configuration

	Running	Name	Reconciliation	Allowed		Id
	☐	Synchronization	✓	✓	▶	dbaeb1e

See the log

If we want to make sure that our Identities were made, we can look into our identities or we can look into the log of our **Synchronization**.

1) Click on the **magnifying glass** and then in the top of the page find **Logs** and follow to this page.



Synchronization details

Settings | Specific settings | Filter | Logs

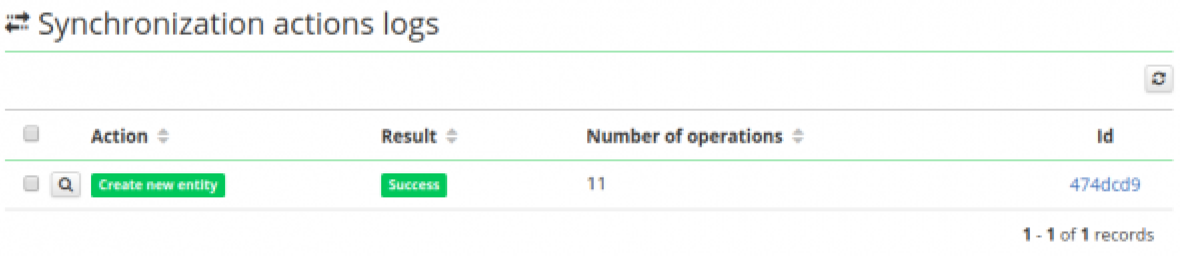
Synchronization logs

	Running	Results	Started	Finished	Id
	☐	11 Created entities	09.03.2018 11:22:11	09.03.2018 11:22:14	4c9024d

1 - 1 of 1 records

2) Now the result should say: "Some number" Created Entities as shown in the picture (we created 11 identities).

3) We can go further and finds out which identities were created. Just click again on the magnifying glass.

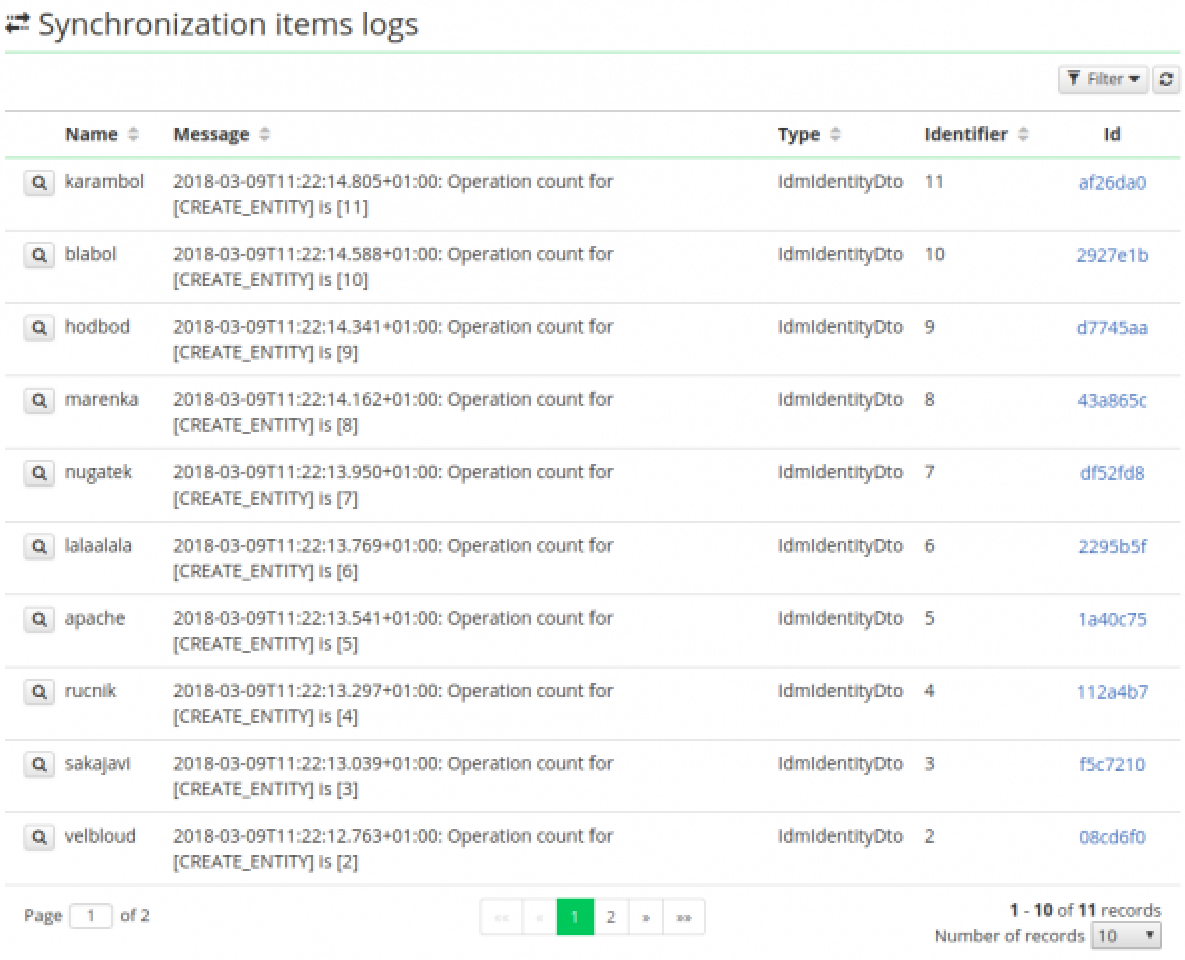


Synchronization actions logs

	Action	Result	Number of operations	Id
	Create new entity	Success	11	474dcd9

1 - 1 of 1 records

4) You will see the log in the top part of the page but we will look at the bottom as the picture shows. There click on the magnifying glass.



Name	Message	Type	Identifier	Id
karambol	2018-03-09T11:22:14.805+01:00: Operation count for [CREATE_ENTITY] is [11]	IdmIdentityDto	11	af26da0
blabol	2018-03-09T11:22:14.588+01:00: Operation count for [CREATE_ENTITY] is [10]	IdmIdentityDto	10	2927e1b
hodbod	2018-03-09T11:22:14.341+01:00: Operation count for [CREATE_ENTITY] is [9]	IdmIdentityDto	9	d7745aa
marenka	2018-03-09T11:22:14.162+01:00: Operation count for [CREATE_ENTITY] is [8]	IdmIdentityDto	8	43a865c
nugatek	2018-03-09T11:22:13.950+01:00: Operation count for [CREATE_ENTITY] is [7]	IdmIdentityDto	7	df52fd8
lalaalala	2018-03-09T11:22:13.769+01:00: Operation count for [CREATE_ENTITY] is [6]	IdmIdentityDto	6	2295b5f
apache	2018-03-09T11:22:13.541+01:00: Operation count for [CREATE_ENTITY] is [5]	IdmIdentityDto	5	1a40c75
rucnik	2018-03-09T11:22:13.297+01:00: Operation count for [CREATE_ENTITY] is [4]	IdmIdentityDto	4	112a4b7
sakajavi	2018-03-09T11:22:13.039+01:00: Operation count for [CREATE_ENTITY] is [3]	IdmIdentityDto	3	f5c7210
velbloud	2018-03-09T11:22:12.763+01:00: Operation count for [CREATE_ENTITY] is [2]	IdmIdentityDto	2	08cd6f0

Page 1 of 2 — 1 2 — 1 - 10 of 11 records — Number of records 10

5) Now you should see the same as is shown in the picture.

This is the basic usage of this connector and now you can try also filters. Just set all filters by your choice and synchronize! To be able to use filters follow please to this tutorial.

Provisioning

Provisioning is not supported yet!

Potential issues

When testing the connector you can see get an error "Connection test failed: Can't read given path. Check if can be read or if it is right!" If that happens make sure your csv file is in a directory to which tomcat has permission to read. You can move the file to the czechidm/data folder which should solve the issue.