univention
be open.

UCS@school

Handbuch für den grafischen Benutzer-Import

Version 5.0
Stand: 1. September 2021

Univention GmbH
Mary-Somerville-Straße 1
28359 Bremen
Deutschland
feedback@univention.de

univention
be open.

Inhaltsverzeichnis

Kapitel 1. Zielgruppe

Dieses Handbuch richtet sich an Mitarbeiter, die den grafischen Import von Benutzern durchführen, und ab Kapitel 4 an Administratoren, die ihn installieren und konfigurieren.

Kapitel 2. Einführung

UCS@school bringt seit der Version 4.2 v6 ein UMC-Modul mit, das es ermöglicht, sicher und komfortabel Benutzerdaten aus CSV-Dateien zu importieren. Über ein flexibles Sicherheitskonzept kann einzelnen Benutzern oder ganzen Gruppen die Berechtigung gegeben werden, Importe für bestimmte Schulen durchführen und deren Ergebnisse einsehen zu können.

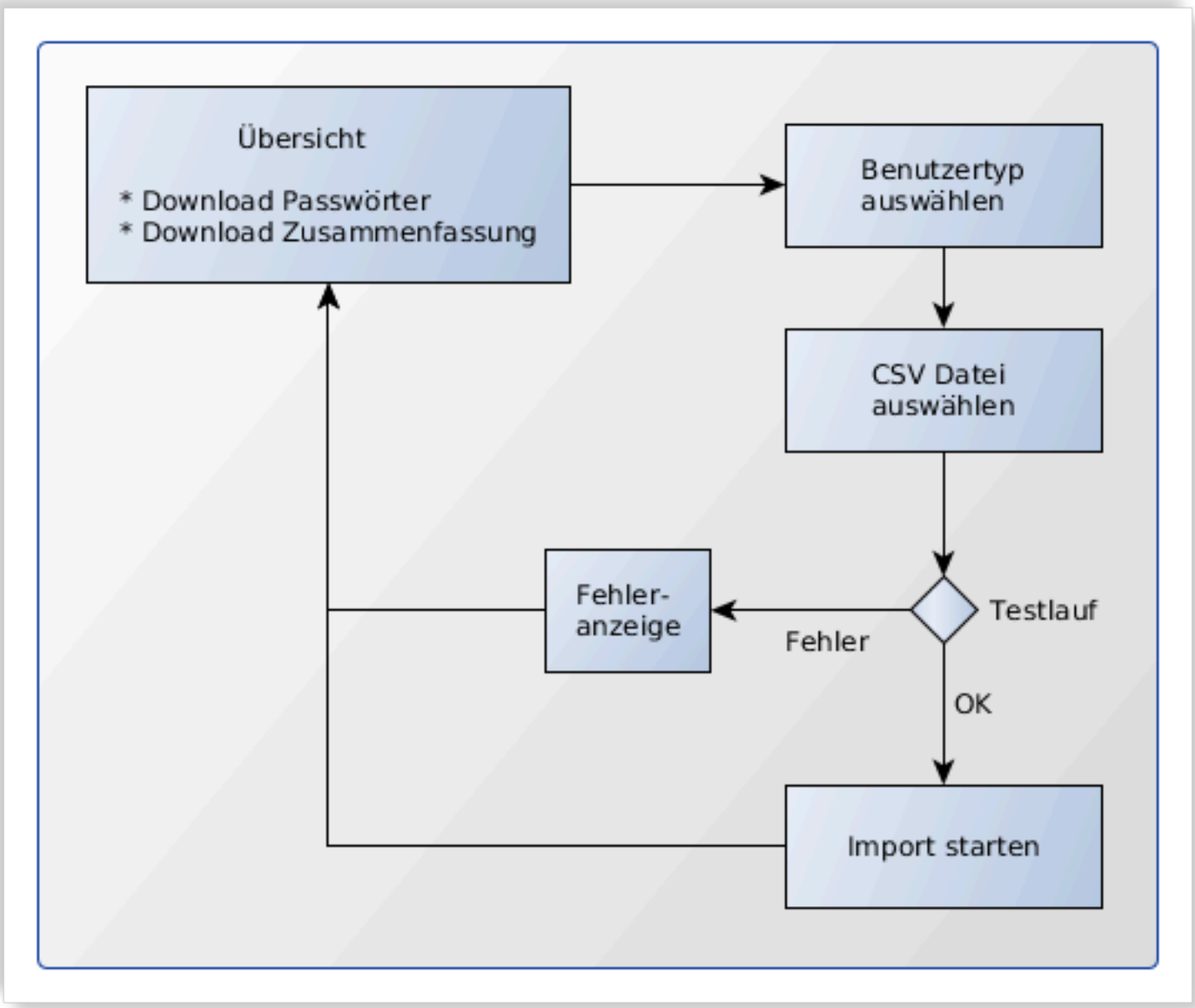
Technisch basiert das UMC-Modul **Benutzerimport** auf Komponenten der Software, die in *UCS@school-Handbuch zur CLI-Import-Schnittstelle*[1] beschrieben sind. Die Konfiguration dieser Komponenten ist nicht Teil dieses Dokuments.

[1] https://docs.software-univention.de/ucsschool-import-handbuch-5.0.html

Kapitel 3. Ablauf des Importvorgangs

Das UMC-Modul leitet den Anwender in mehreren Schritten durch den Import:

Abbildung 3.1. Schritte eines Importvorganges

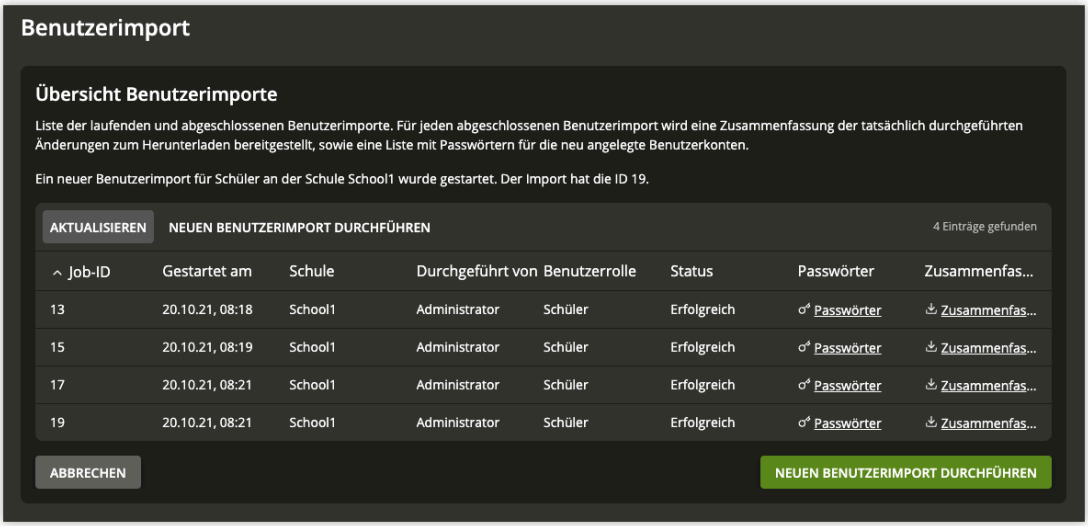


Ein neuer Import kann in der Übersichtsseite durch Klicken auf "Neuen Benutzerimport durchführen" gestartet werden. Wenn noch nie ein Import durchgeführt wurde, startet das UMC-Modul direkt mit den erstem Schritt für einen neuen Import. In allen anderen Fällen wird zunächst die Übersicht angezeigt.

Anmerkung

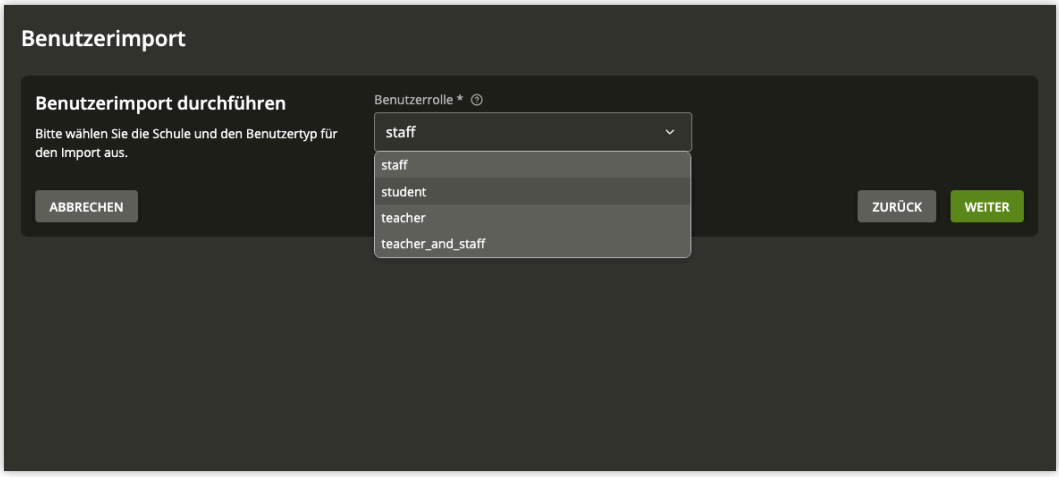
Sollte sich der Anwender per SAML (*Single Sign-On*) angemeldet haben, erscheint ein Fenster, das (u.U. mehrfach) zur Eingabe des eigenen Benutzerpasswortes auffordert.

Abbildung 3.2. Übersichtsseite



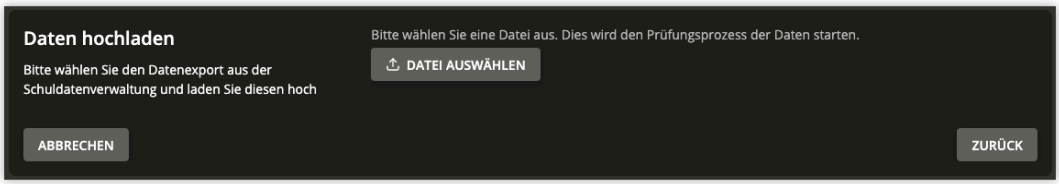
1. Zuerst muss der Typ der zu importierenden Benutzer ausgewählt werden.

Abbildung 3.3. Auswahl des Benutzertyps



2. Anschließend kann die CSV-Datei mit den Benutzerdaten ausgewählt werden.

Abbildung 3.4. Hochladen der CSV-Datei

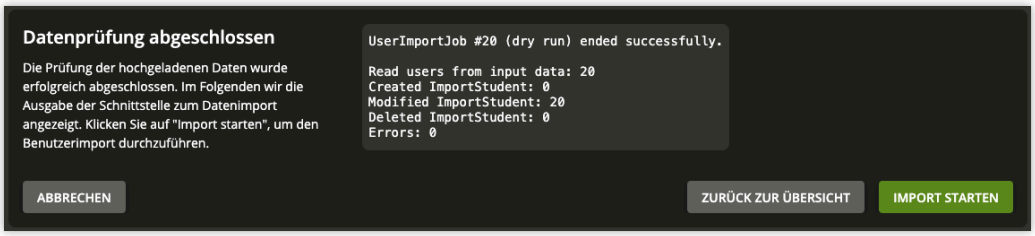


3. Nun werden die Daten geprüft und es wird ein Test-Import durchgeführt, um mögliche Fehler vorab zu erkennen. Das Benutzerverzeichnis wird dabei nicht verändert.

4. Je nach Menge der zu importierenden Daten, kann der Test-Import einige Zeit beanspruchen.

- War die Simulation erfolgreich, kann nun der tatsächlich Import gestartet werden.

Abbildung 3.5. Simulation war erfolgreich

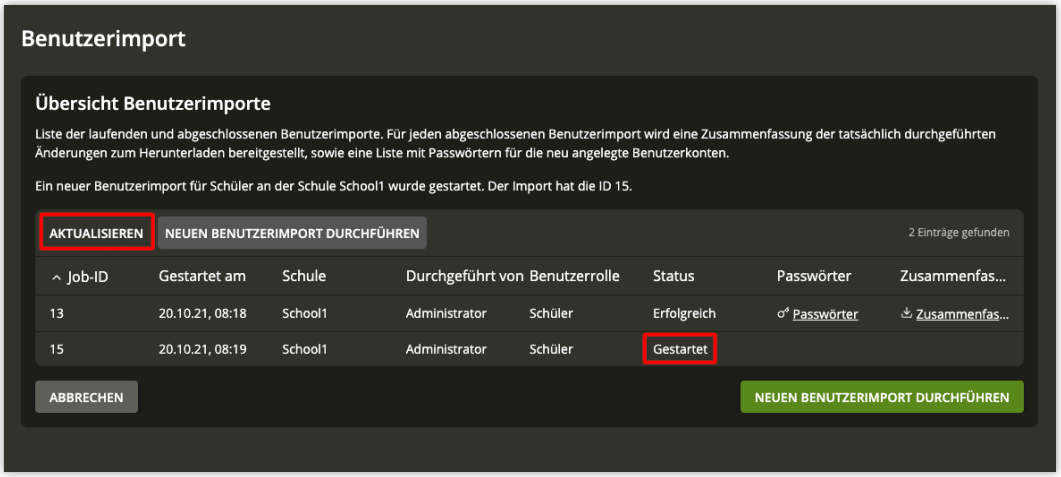


- Traten während des Test-Imports Fehler auf, wird eine Fehlermeldung angezeigt. Unterhalb der Fehlermeldung ist im Text ein Link. Durch Klicken auf diesen, wird eine E-Mail mit der Fehlermeldung an einen Administrator verfasst.

Abbildung 3.6. Simulation hatte Fehler

5. Nach dem Start des Imports kehrt das UMC-Modul zur Übersichtsseite zurück. Wenn der neue Import-Job noch nicht angezeigt wird, kann die Liste mit der Schaltfläche "Aktualisieren" neu geladen werden.

Abbildung 3.7. Übersichtsseite mit gestartetem Import



Kapitel 4. Installation, Konfiguration und Dateiformat

4.1. Installation

Feedback

Die Installation muss auf dem Primary Directory Node stattfinden:

```
# univention-install ucs-school-umc-import
```

4.2. Konfiguration

Feedback

Das Setzen der Univention Configuration Registry-Variablen `ucsschool/import/error/mail-address` ist wichtig, damit Anwender beim Auftreten eines Fehlers, eine E-Mail an den Administrator schicken können, indem sie auf den oben beschriebenen Link klicken.

```
# ucr set ucsschool/import/error/mail-address=admin@ihre-schule.de
```

Technisch basiert der grafische Benutzer-Import auf Komponenten der Software die in *Handbuch Import-Schnittstelle*[1] beschrieben sind. Deren Konfiguration erfolgt in einer JSON Datei. Die Datei `/usr/share/ucs-school-import/configs/user_import_http-api.json` sollte als Ausgangsbasis für eigene, angepasste Konfigurationen verwendet werden. Die Konfiguration wird aktiviert, indem sie an die richtige Position kopiert wird:

```
# cp /usr/share/ucs-school-import/configs/user_import_http-api.json \
   /var/lib/ucs-school-import/configs/user_import.json
```

Das Sicherheitskonzept ermöglicht es Benutzern Rechte zu erteilen, um Importe nur an bestimmten Schulen und nur für bestimmte Benutzertypen durchführen, sowie die Ergebnisse dieser Import-Jobs einzusehen. Während der Installation wurde für jede Schule eine Gruppe *$OU*`-import-all` erstellt. An diesen Gruppen wurde die Option `UCS@school Import-Berechtigungen` aktiviert. In der UMC können für diese Gruppen auf der Karteikarte **UCS@school** `Import-Berechtigungen` festgelegt werden.

Eine `Import-Berechtigung` setzt sich zusammen aus einer Liste von Schulen (standardmäßig nur die Schule für die die Gruppe erzeugt wurde) und einer Liste von Benutzertypen (Rollen). Alle Benutzer die Mitglieder dieser Gruppe sind können Imports für die aufgelisteten Benutzertypen and den aufgelisteten Schulen durchführen. Verschachtelte Gruppen werden nicht unterstützt.

Sollen zusätzlich zu den automatisch erzeugten Gruppen neue angelegt werden, so muss an diesen zum einen die Option `UCS@school Import-Berechtigungen` aktiviert, und zum anderen die UMC-Richtlinie `cn=schoolimport-all,cn=UMC,cn=policies,`*`$LDAP_BASE`* zugewiesen werden.

Alle an einem Import-Job beteiligten, und von ihm erzeugten, Dateien finden sich unter `/var/lib/ucs-school-import/jobs/`*`$JAHR/$JOB-ID/`*: Konfigurationsdateien, Hooks, Logdateien, CSV-Dateien (Eingabedaten, Passwörter neuer Benutzer, Zusammenfassung).

[1] https://docs.software-univention.de/ucsschool-import-handbuch-5.0.html

Anmerkung

Sollte auf dem Primary Directory Node ein SSL-Zertifikat mit abweichenden FQDNs verwendet werden, wird beim Öffnen des UMC-Moduls **Benutzerimport** eine Fehlermeldung auftauchen, da der lokale Rechnername nicht mit den Rechnernamen im SSL-Zertifikat übereinstimmt. In diesem Fall muss die UCR-Variable `ucsschool/import/http_api/client/server` entsprechend auf den/einen Rechnernamen (FQDN) des SSL-Zertifikats gesetzt werden. Zusätzlich sollte die UCR-Variable `ucsschool/import/http_api/ALLOWED_HOSTS` den lokalen FQDN sowie den im SSL-Zertifikat verwendeten FQDN enthalten. Nach dem Setzen der beiden UCR-Variablen müssen einige Dienste neu gestartet werden: `systemctl restart ucs-school-import-http-api ucs-school-import-celery-worker`.

4.3. Datenformat

Feedback

Das Format der CSV-Datei ist anpassbar. Generell gilt aber folgendes:

- Die erste Zeile führt die Bezeichner der Spalten auf. Zum Beispiel:

  ```
  "Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
  ```

- Daten in Spalten sind in doppelten Anführungszeichen eingeschlossen.
- Die Spalten sind durch Komma voneinander getrennt.
- Es muss jeweils eine Spalte für die primäre Schule eines Benutzers, seinen Vor- und Nachnamen geben.
- Mehrere Klassennamen werden durch Komma, ohne Freizeichen, getrennt aufgezählt (z.B. `1a,2b,3c`). Klassennamen dürfen, aber brauchen nicht, den Namen der Schule (mit einem Bindestrich verbunden) vorangestellt haben (z.B. `Scholl-1a,Scholl-2b,Scholl-3c`). Wird der Name der Schule vorangestellt, *muss* dies der gleiche Wert sein wie in der Spalte für die Schule.

Beispieldaten für Testläufe können mit Hilfe eines Skripts erzeugt werden:

```
# /usr/share/ucs-school-import/scripts/ucs-school-testuser-import \
   --httpapi \      # Format passend zu user_import_http-api.json
 erzeugen
   --students 20 \  # Anzahl Benutzer, alternativ: --staff --teachers --
staffteachers
   --classes 2 \                  # Anzahl zu erzeugender Klassen
   --create-email-addresses \     # E-Mail-Adressen erzeugen
   SchuleEins                     # Schule (OU) in die importiert
 werden soll
```

Die erzeugte Datei heißt `test_users_`*`$DATUM_$UHRZEIT`*`.csv` und passt zur Konfiguration in `/usr/share/ucs-school-import/configs/ucs-school-testuser-http-import.json`.

Eine solche Datei sieht z.B. so aus:

```
"Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
"SchuleEins","Jeanne","Oberbockstruck","1a","A
 student.","+24-165-622645","jeannem.oberbockstruck@example.de"
"SchuleEins","Jehanne","Obergöker","1b","A
 student.","+16-456-810331","jehannem.mobergoeker@example.de"
"SchuleEins","Çetin","Schrage","1a","A
 student.","+93-982-722661","cetinm.schrage@example.de"
```

```
"SchuleEins","Zwenna","Schomaker","1b","A
 student.","+39-504-246300","zwennam.schomakerm@example.de"
```

Kapitel 5. Test an der Kommandozeile

Das Testen einer Konfiguration, insbesondere bei Änderungen am *Spalten-Mapping*, ist u.U. an der Kommandozeile schneller als in der UMC. Bei Verwendung der richtigen Kommandozeilenparameter wird *beinahe* der gleiche Importvorgang ausgeführt, wie wenn er vom UMC-Modul gestartet würde.

Das Skript, das Beispieldaten erzeugt, druckt am Ende die benötigten Kommandozeilenparameter exakt aus. Hier ein Beispiel:

```
--school 'SchuleEins' --user_role 'ROLE' --source_uid 'schuleeins-ROLE'
 \
--conffile '/usr/share/ucs-school-import/configs/ucs-school-testuser-
http-import.json' \
--infile 'test_users_2018-07-04_12:31:46.csv'
```

ROLE muss mit `student`, `staff`, `teacher` oder `teacher_and_staff` ersetzt werden, und `SchuleEins` mit der entsprechenden *OU* (in `'schuleeins-ROLE'` in Kleinbuchstaben).

Kapitel 6. Szenario "Eine Quelle, partielle Importe" (*single source, partial import - SiSoPi*)

Seit UCS@school 4.3 v5 wird ein Szenario unterstützt, in dem es eine Quelldatenbank mit den Mitgliedern aller Schulen gibt, bei dem aber nicht zentral, sondern an allen Schulen einzeln importiert wird. Der Import ist sowohl über die Kommandozeile als auch mit dem UMC-Modul (welches im Hintergrund die *Import-HTTP-API* verwendet) möglich.

6.1. Features

Feedback

- *OU-übergreifende* Benutzerkonten (ein Benutzer kann in mehreren Schulen sein)
- Jede Schule kann ihre Benutzer einzeln und zu einem beliebigen Zeitpunkt importieren.

6.2. Voraussetzungen

Feedback

- Eine Datenbasis, die alle Benutzer mit je einem domänenweit eindeutigen Schlüssel (`record_uid`), enthält.
- Die Quelldatenbank exportiert separate CSV Dateien pro Schule und Benutzerrolle.
- Die Importe können in zufälliger Reihenfolge stattfinden. Es ist es möglich, dass beim Verschieben eines Benutzers dieser zuerst in einer Schule gelöscht und in einer Anderen später angelegt wird. Das Benutzerkonto darf in der Zwischenzeit nicht gelöscht werden.

6.3. Implementierung

Feedback

Um das Verschieben eines Benutzers von Schule A nach Schule B in zwei Schritten zu ermöglichen - einschließlich der Möglichkeit, dass der Benutzer zuerst in Schule A gelöscht und später in Schule B angelegt wird - wird eine temporäre Schule verwendet: die sog. `limbo_ou`. Es handelt sich dabei um eine gewöhnliche Schule (`OU`), deren Name konfigurierbar ist (Standard ist `limbo`).

Benutzerkonten, die von ihrer letzten bzw. einzigen Schule ("A") entfernt wurden, werden 1. sofort deaktiviert und 2. in die temporäre Schule (`limbo_ou`) verschoben.

Soll ein Benutzer während eines Imports (an Schule "B") erstellt werden, existiert jedoch bereits ein Konto mit dessen `record_uid` in der "*limbo* OU", so wird dieses Konto stattdessen von dort zur Schule "B" verschoben und das Konto reaktiviert.

6.4. Installation und Konfiguration

Feedback

Der Inhalt von `/usr/share/ucs-school-import/configs/user_import_sisopi.json` muss der Importkonfiguration (in `/var/lib/ucs-school-import/configs/user_import.json`) hinzugefügt werden.

Folgende Einstellungen sollten angepasst werden:

- `deletion_grace_period:deactivation` *muss* `0` sein.
- `deletion_grace_period:deletion` sollte (deutlich) größer als `0` sein. Es sollte die maximale Anzahl an Tagen sein, die ein Import von zwei Schulen auseinander liegen kann. Das ist die Zeit die ein Konto in der "*limbo* OU" verbringt, bevor es endgültig gelöscht wird.

Der Name der sog. "*limbo* OU" kann mit der Einstellung `limbo_ou` gesetzt werden.

Darüber hinaus muss die Univention Configuration Registry-Variable `ucsschool/import/http_api/set_source_uid` auf `no` gesetzt und der Import-HTTP-API-Server neu gestartet werden:

```
# ucr set ucsschool/import/http_api/set_source_uid=no
# service ucs-school-import-http-api restart
```

6.5. Beispielaufbau

Für den Testaufbau werden zunächst zwei reguläre und die temporäre Schule erstellt:

```
# /usr/share/ucs-school-import/scripts/create_ou schuleA
# /usr/share/ucs-school-import/scripts/create_ou schuleB
# /usr/share/ucs-school-import/scripts/create_ou limbo
```

Nach dem Sichern der ursprünglichen Konfiguration wird die *SiSoPi-Konfiguration* aktiviert. Üblicherweise wird die neue Konfiguration anschließend an die individuellen Erfordernisse angepasst. Für den Testaufbau wurden `csv`, `scheme` und `source_uid` hinzugefügt.

```
# cp -v /var/lib/ucs-school-import/configs/user_import.json{,.bak}
# cp -v /usr/share/ucs-school-import/configs/user_import_sisopi.json /
var/lib/ucs-school-import/configs/user_import.json
# $EDITOR /var/lib/ucs-school-import/configs/user_import.json
```

Tipp: Mit folgendem Befehl kann die syntaktische Korrektheit der JSON-Datei geprüft werden. Wenn die Datei syntaktisch korrekt ist, wird ihr Inhalt ausgegeben, bei einem Fehler wird stattdessen dieser angezeigt.

```
# cat /var/lib/ucs-school-import/configs/user_import.json | python3 -m
 json.tool
{
    "classes": {
        "user_importer":
 "ucsschool.importer.mass_import.sisopi_user_import.SingleSourcePartialUserImport"
    },
    "configuration_checks": [
        "defaults",
        "sisopi"
    ],
    "csv": {
        "mapping": {
            "Beschreibung": "description",
            "EMail": "email",
            "Klassen": "school_classes",
            "Nachname": "lastname",
            "Schule": "school",
            "Telefon": "phone",
            "Vorname": "firstname"
```

```
        }
    },
    "deletion_grace_period": {
        "deactivation": 0,
        "deletion": 90
    },
    "limbo_ou": "limbo",
    "scheme": {
        "record_uid": "<firstname>.<lastname>",
        "username": {
            "default":
 "<:umlauts><firstname>.<lastname><:lower>[COUNTER2]"
        }
    },
    "source_uid": "Test"
}
```

```
# ucr set ucsschool/import/http_api/set_source_uid=no
# service ucs-school-import-http-api restart
```

Nun wird für jede Schule eine zu importierende CSV Datei erzeugt:

```
# /usr/share/ucs-school-import/scripts/ucs-school-testuser-import \
 --csvfile test_users_A-1.csv --nostart --httpapi --teachers 4 \
 --classes 1 --inclasses 1 --schools 1 --verbose schuleA
# /usr/share/ucs-school-import/scripts/ucs-school-testuser-import \
 --csvfile test_users_B-1.csv --nostart --httpapi --teachers 4 \
 --classes 1 --inclasses 1 --schools 1 --verbose schuleB
```

```
# cat test_users_A-1.csv

"Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
"schuleA","Yola","Lenz","1a","A teacher.","+74-686-445678",""
"schuleA","Iphigenie","Lemgo","1a","A teacher.","+63-727-768248",""
"schuleA","Felix","Adams","1a","A teacher.","+15-263-530094",""
"schuleA","Radomila","Meygger","1a","A teacher.","+11-364-599925",""
```

```
# cat test_users_B-1.csv

"Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
"schuleB","Stan","Kinker","1a","A teacher.","+91-299-143803",""
"schuleB","Jonathan","Heuelmann","1a","A teacher.","+74-656-351455",""
"schuleB","Ingward","Bohnenkämper","1a","A teacher.","+24-351-217608",""
"schuleB","Vincent","Störtländer","1a","A teacher.","+67-303-103581",""
```

Der Import würde regulär über das UMC-Modul statt finden, wird für diesen Test aber an der Kommandozeile durchgeführt. Beim Import an den beiden Schulen werden je vier Lehrer angelegt:

```
# /usr/share/ucs-school-import/scripts/ucs-school-user-import \
 --verbose --user_role teacher --infile test_users_A-1.csv \
 --school schuleA

------ User import statistics ------
Read users from input data: 4
Created ImportTeacher: 4
  ['yola.lenz', 'iphigenie.lemgo', 'felix.adams', 'radomila.meygger']
```

```
Modified ImportTeacher: 0
Deleted ImportTeacher: 0
Errors: 0
------ End of user import statistics ------

# /usr/share/ucs-school-import/scripts/ucs-school-user-import \
 --verbose --user_role teacher --infile test_users_B-1.csv \
 --school schuleB

------ User import statistics ------
Read users from input data: 4
Created ImportTeacher: 4
  ['stan.kinker', 'jonathan.heuelman', 'ingward.bohnenkae',
 'vincent.stoertlae']
Modified ImportTeacher: 0
Deleted ImportTeacher: 0
Errors: 0
------ End of user import statistics ------
```

Nun soll `yola.lenz` von `schuleA` nach `schuleB` verschoben werden. Dazu wird eine CSV Datei `test_users_A-2.csv` ohne die Zeile mit `"Yola","Lenz"` aus `test_users_A-1.csv` erzeugt, sowie eben diese Zeile in `test_users_B-2.csv` eingefügt. Dort muss `schuleA` noch durch `schuleB` ersetzt werden. Die neuen Dateien sehen wie folgt aus:

```
# cat test_users_A-2.csv

"Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
"schuleA","Iphigenie","Lemgo","1a","A teacher.","+63-727-768248",""
"schuleA","Felix","Adams","1a","A teacher.","+15-263-530094",""
"schuleA","Radomila","Meygger","1a","A teacher.","+11-364-599925",""
```

```
# cat test_users_B-2.csv

"Schule","Vorname","Nachname","Klassen","Beschreibung","Telefon","EMail"
"schuleB","Stan","Kinker","1a","A teacher.","+91-299-143803",""
"schuleB","Jonathan","Heuelmann","1a","A teacher.","+74-656-351455",""
"schuleB","Ingward","Bohnenkämper","1a","A teacher.","+24-351-217608",""
"schuleB","Vincent","Störtländer","1a","A teacher.","+67-303-103581",""
"schuleB","Yola","Lenz","1a","A teacher.","+74-686-445678",""
```

Beim Import an `schuleA` wird `yola.lenz` scheinbar gelöscht. Tatsächlich wird sie aber in die Schule `limbo` verschoben:

```
# udm users/user list --filter uid=yola.lenz | egrep 'DN|school:'
DN: uid=yola.lenz,cn=lehrer,cn=users,ou=schuleA,<base dn>
  school: schuleA

# /usr/share/ucs-school-import/scripts/ucs-school-user-import \
 --verbose --user_role teacher --infile test_users_A-2.csv \
 --school schuleA

[..]
------ Deleting 1 users... ------
Removing ImportTeacher(name='yola.lenz', school='schuleA',
 dn='uid=yola.lenz,cn=lehrer,cn=users,ou=schuleA,<base dn>') from school
 'schuleA'...
```

```
Moving ImportTeacher(name='yola.lenz', school='schuleA',
 dn='uid=yola.lenz,cn=lehrer,cn=users,ou=schuleA,<base dn>') to limbo
 school u'limbo'.
[..]

------ User import statistics ------
Read users from input data: 3
Modified ImportTeacher: 3
  ['iphigenie.lemgo', 'felix.adams', 'radomila.meygger']
Deleted ImportTeacher: 1
  ['yola.lenz']
Modified ImportTeacher: 0
Deleted ImportTeacher: 0
Errors: 0
------ End of user import statistics ------

# udm users/user list --filter uid=yola.lenz | egrep 'DN|school:'
DN: uid=yola.lenz,cn=lehrer,cn=users,ou=limbo,<base dn>
  school: limbo
```

Beim Import an `schuleB` wird `yola.lenz` aus der Schule `limbo` dort hin verschoben:

```
# /usr/share/ucs-school-import/scripts/ucs-school-user-import \
 --verbose --user_role teacher --infile test_users_B-2.csv \
 --school schuleB

[..]
User ImportTeacher(name='yola.lenz', school='limbo',
 dn='uid=yola.lenz,cn=lehrer,cn=users,ou=limbo,<base dn>') is in limbo
 school u'limbo', moving to 'schuleB'.
Reactivating ImportTeacher(name=None, school='schuleB', dn=None)...
User will change school. Previous school: 'limbo', new school:
 'schuleB'.
Moving ImportTeacher(name='yola.lenz', school='limbo',
 dn='uid=yola.lenz,cn=lehrer,cn=users,ou=limbo,<base dn>') from school
 'limbo' to 'schuleB'...
[..]

------ User import statistics ------
Read users from input data: 5
Modified ImportTeacher: 5
  ['stan.kinker', 'jonathan.heuelman', 'ingward.bohnenkae',
 'vincent.stoertlae']
  ['yola.lenz']
Modified ImportTeacher: 0
Deleted ImportTeacher: 0
Errors: 0
------ End of user import statistics ------

# udm users/user list --filter uid=yola.lenz | egrep 'DN|school:'
DN: uid=yola.lenz,cn=lehrer,cn=users,ou=schuleB,<base dn>
  school: schuleB
```

Der umgekehrte Fall, in dem ein zu verschiebender Benutzer an der Zielschule importiert wird, bevor er an der ursprünglichen Schule gelöscht wird, kann z.B. folgendermaßen erzeugt werden: Die Zeile von `"Iphi-`

genie","Lemgo" wird in das CSV der schuleB kopiert, wobei die Spalte "Schule" angepasst und aus dem CSV der schuleA entfernt wird. Der Import wird nun an schuleB vor schuleA durchgeführt. Zwischendurch wird die Lehrerin Mitglied beider Schulen sein. Das Benutzerkonto würde sich folgendermaßen ändern:

```
# vor dem Import:
DN: uid=iphigenie.lemgo,cn=lehrer,cn=users,ou=schuleA,<base dn>
  school: schuleA

# nach dem Import an schuleB:
DN: uid=iphigenie.lemgo,cn=lehrer,cn=users,ou=schuleA,<base dn>
  school: schuleA
  school: schuleB

# nach dem Import an schuleA:
DN: uid=iphigenie.lemgo,cn=lehrer,cn=users,ou=schuleB,<base dn>
  school: schuleB
```