

# Reprocess all files for a given user

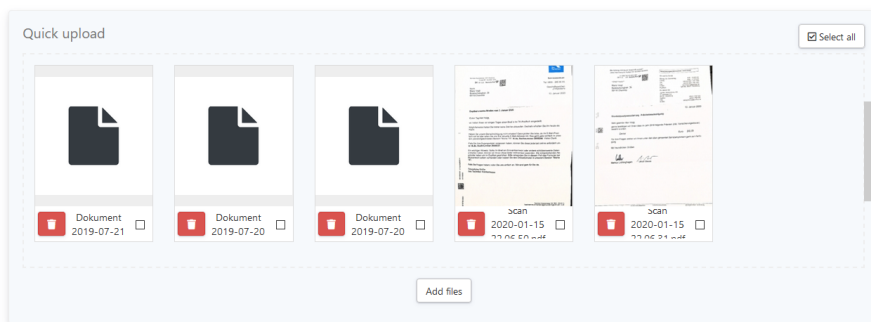
- Note that you can only re-process files which are your's! If you want to reprocess everything for everybody you will need to loop over each user while getting his username/password or auth\_token (only way if user has 2FA secured account) from database. An auth\_token can only be stripped out from database if the user did not log off (otherwise it's null/empty).
- reprocessing makes sense in case of updating tesseract version which might improve the OCR text output quality. So for longterm storing of documents it might be worth to reprocess files.
- Reprocessing will raise a lot of background tasks into schedule:

## Background tasks

There is currently 285 queued tasks. 🕒

File processing, thumbnail creation, index update, optical character recognition are background tasks. A large amount of unprocessed tasks will result in incomplete search results.

- Note: Reprocessing of files does not work in quick upload mask (only works for files assigned to existing documents):



## Way 1: pure API calls ? for a regular user (no 2FA secured)

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
TEEDY_USER="admin"
TEEDY_USER_PASS="password"
AUTH_TOKEN=$(curl -i -X POST -d username="$TEEDY_USER" -d password="$TEEDY_USER_PASS"
"$BASE_URL/api/user/login" -k|grep "auth_token"|cut -c24-59)
BACKUP_DIR="/backup/teedy"
TARGET_DOCLIST_JSON=$BACKUP_DIR"/documentlist_forfiles.json"
TARGET_FILELIST_JSON=$BACKUP_DIR"/filelist.json"
mkdir -p "$BACKUP_DIR"
rm $TARGET_DOCLIST_JSON
rm $TARGET_FILELIST_JSON
```

```

echo "Retrieving document list"
curl --silent -X GET -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/document/list?limit=0"
-k | jq . > "$TARGET_DOCLIST_JSON"
echo "Retrieving file list based on document list"
COUNT=0
jq -c '.[{documents}].[[]].[[]]{id}+{title}+{create_date}' "$TARGET_DOCLIST_JSON" | while read -
r i; do
    COUNT=$((COUNT + 1))
    DOC_ID=$(jq -c '.[{id}].id' <<< $(printf '%s\n' "$i"))
    DOC_ID=${DOC_ID:1:-1}
    curl --silent -X GET -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/list?id=$DOC_ID" >> "$TARGET_FILELIST_JSON"
    echo Getting $COUNT : $DOC_ID
    #put some sleep time here if your server has less ressources. Otherwise you might overload
PostgreSQL as well as Jetty. It leads to unstability of the running instance
    #have a look at https://dms.yourdomain.de/#/settings/monitoring to check the average OCR
time per document. Usually 3 to 10 seconds should be normal
    #sleep 5
done
echo "Starting to process files"
COUNT=0
jq -c '.[[]].[[]]{id}' "$TARGET_FILELIST_JSON" | while read -r i; do
    COUNT=$((COUNT + 1))
    FILE_ID=$(jq -c '.[{id}].id' <<< $(printf '%s\n' "$i"))
    FILE_ID=${FILE_ID:1:-1}
    echo Processing $COUNT : $FILE_ID
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
done
curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/user/logout" -k

```

## Way 2: API + psql calls ? for a regular user (no 2FA secured)

This script is much shorter and more elegant to reprocess. Note that this script logs in the user by a fresh created token. It will fail if the user login is secured with 2FA. In this case see below!

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USER="admin"
TEEDY_USER_PASS="password"
AUTH_TOKEN=$(curl -i -X POST -d username="$TEEDY_USER" -d password="$TEEDY_USER_PASS"
"$BASE_URL/api/user/login" -k|grep "auth_token"|cut -c24-59)

for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F JOIN
t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND
U.use_username_c = '$TEEDY_USER';"); do
    echo PROCESSING "$FILE_ID"
    curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
    sleep 5
done

#logout
curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN" "$BASE_URL/api/user/logout" -k
```

## Way 3: API + psql calls ? for a single 2FA secured user

The following script can be used if your user is secured with 2FA. Note that the user needs to be logged in once (if he logs off the recent token will be destroyed).

```
#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USER="admin"

#this reads exactly one token from the given user. If the user is not logged in it will be
null and the token will be null too!
AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
```

```

for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F JOIN
t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND
U.use_username_c = '$TEEDY_USER';"); do
    echo PROCESSING "$FILE_ID"
    #curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
    sleep 2
done

#do not logout to keep the token

```

## Way 4: API + psql calls ? loop over all users (2FA secured users)

```

#!/bin/bash
BASE_URL="https://dms.yourdomain.de"
DB_USER="teedy"
DB_NAME="teedy_db"
TEEDY_USERS=$(psql -t -U$DB_USER $DB_NAME --command="SELECT use_username_c FROM t_user;")

for TEEDY_USER in $TEEDY_USERS; do
    echo LOGGING IN AS $TEEDY_USER
    #this reads exactly one token from the given user. If the user is not logged in it will be
null and the token will be null too!
    AUTH_TOKEN=$(psql -t -U$DB_USER $DB_NAME --command="SELECT aut_id_c FROM
t_authentication_token AS A JOIN t_user AS U ON U.use_id_c = A.aut_iduser_c WHERE
use_username_c = '$TEEDY_USER' AND aut_lastconnectiondate_d IS NOT NULL LIMIT 1;")
    for FILE_ID in $(psql -t -U$DB_USER $DB_NAME --command="SELECT fil_id_c FROM t_file AS F
JOIN t_user AS U ON U.use_id_c = F.fil_iduser_c WHERE F.fil_deletedate_d IS NULL AND
U.use_username_c = '$TEEDY_USER';"); do
        echo PROCESSING "$FILE_ID"
        curl --silent -X POST -H "Cookie: auth_token=$AUTH_TOKEN"
"$BASE_URL/api/file/$FILE_ID/process"
        sleep 2
    done
done

#do not logout after processing (to keep the token alive!)

```

Version #2

Erstellt: 15 Mai 2025 10:07:18 von Mario Voigt

Zuletzt aktualisiert: 15 Mai 2025 10:09:36 von Mario Voigt