ELO server – Installation and operation

Upgrade and migration

Table of contents

Update

ELO server updates

ELO provides updates for ELO server applications on a regular basis. There are several ways to install these updates:

- Use the newest ELO DVD. This updates Java, Apache Tomcat, and the ELO server all at the same time.
- Download and run the newest ELO Server Setup from the ELO SupportWeb. These are released more frequently than DVDs.
- Update individual ELO server applications manually. This is more complicated than the other options, but it ensures that you have the most current version of the application.

Information

In the following paths, replace <ELO> with the directory you installed the ELO ECM Suite to.

All of these update options are described in the following.

General information

Whenever you are updating individual server applications, please read the version notes before proceeding. These inform you if an application is dependent on other applications, if you need to update configuration files, or take other necessary steps.

Update to 21.02 or higher

The update to version 21.02 changes the version of the Elasticsearch used. The update to Elasticsearch 7.15 requires the Elasticsearch index to be reindexed. The ELO Server Setup deletes existing indexes during the update.

Plan more time than is usually required for an update for the reindex.

Alternative: For larger systems (with more than 500,000 documents), create a copy of the production system and reindex there.

For more information, refer to the Upgrade index and Background reindex documentation.

Update to 10.02 or higher

Since ELO 10.2, network traffic between ELOix and ELO iSearch is encrypted by default. If there are multiple ELO iSearch nodes, communication between the nodes is also encrypted. The certificates required for this are generated by the ELO Server Setup during installation, assuming they are not already available. There is a root certificate and a server certificate signed by the root certificate for each node.

These certificates are stored in the *elosetup.conf* file. This file is in:

<ELO>/config/serversetup2/

To enable encrypted communication between these nodes, their certificates must be signed by the same root certificate. If you have ELOenterprise systems with multiple servers, you need to install this root certificate on all servers. The procedure for updating to 10.2 or higher is therefore as follows:

1. Carry out the update on the leading system.
2. Transfer the *elosetup.conf* file from the leading system to the secondary systems.

   (Path: *<ELO>/config/serversetup2/*)
3. Update the components on the secondary systems.

Via the ELO Server Setup

The easiest way to update the server files is to run the ELO Server Setup. This can also be used to update between major release versions.

Please note

When you perform an update in this way, the ELO server will be stopped for a short time. The repository may not be used during this time.

Please note

Updating ELO iSearch may take up a significant amount of disk space if re-indexing is required.

Re-indexing ELO iSearch may require a significant amount of time and resources if your repository is large.

For more information on ELO iSearch, refer to the *ELO iSearch* documentation.

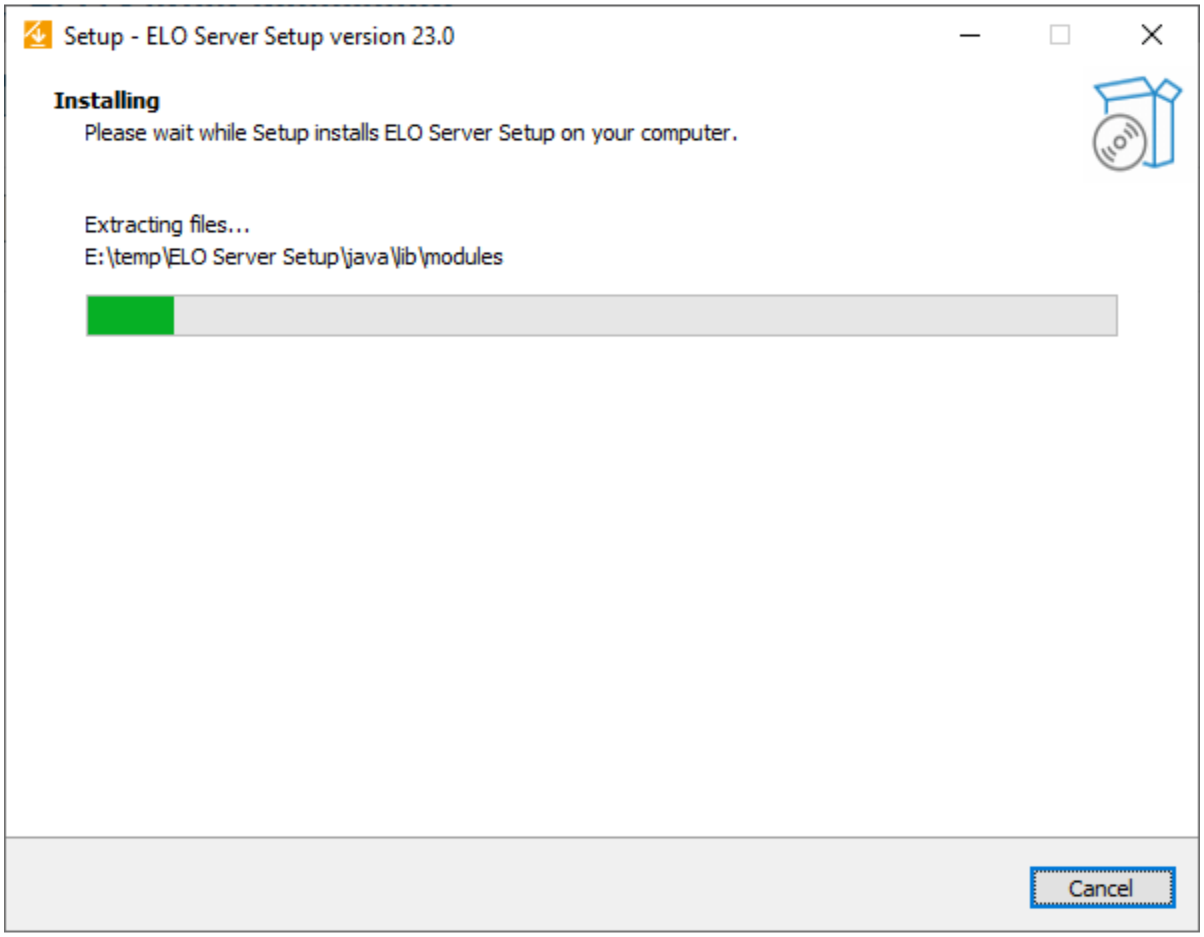
General preparation: Back up your existing ELOprofessional installation. It is easiest to back up the entire ELOprofessional install directory (e. g. *C:\ELOprofessional*) and all child directories. Also back up the databases for the ELO Access Manager and all repositories.

Important

ELO ECM Suite 10 and higher is only compatible with 64-bit operating systems. If your ELO server is currently running on a 32-bit platform, migrate the server to a 64-bit operating system before carrying out the update.

1.

Run the ELO ECM Suite server setup from the ELO DVD or download and run it from the ELO SupportWeb.


Setup - ELO Server Setup version 23.0

Installing

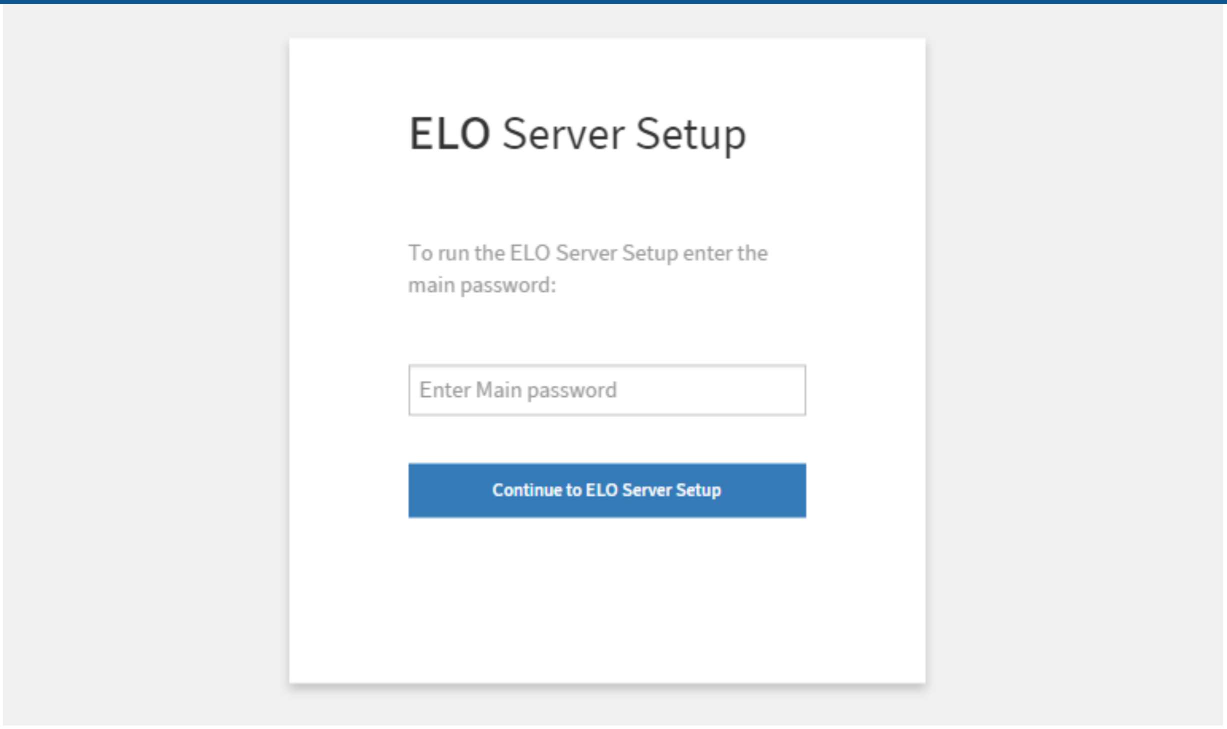
Please wait while Setup installs ELO Server Setup on your computer.

Extracting files...

E:\temp\ELO Server Setup\java\lib\modules

The ELO Server Setup extracts the other setup components to a local temporary directory.

The ELO Server Setup user interface opens automatically in the standard browser.



To run the ELO Server Setup enter the main password:

The ELO Server Setup will prompt you to enter the main password.

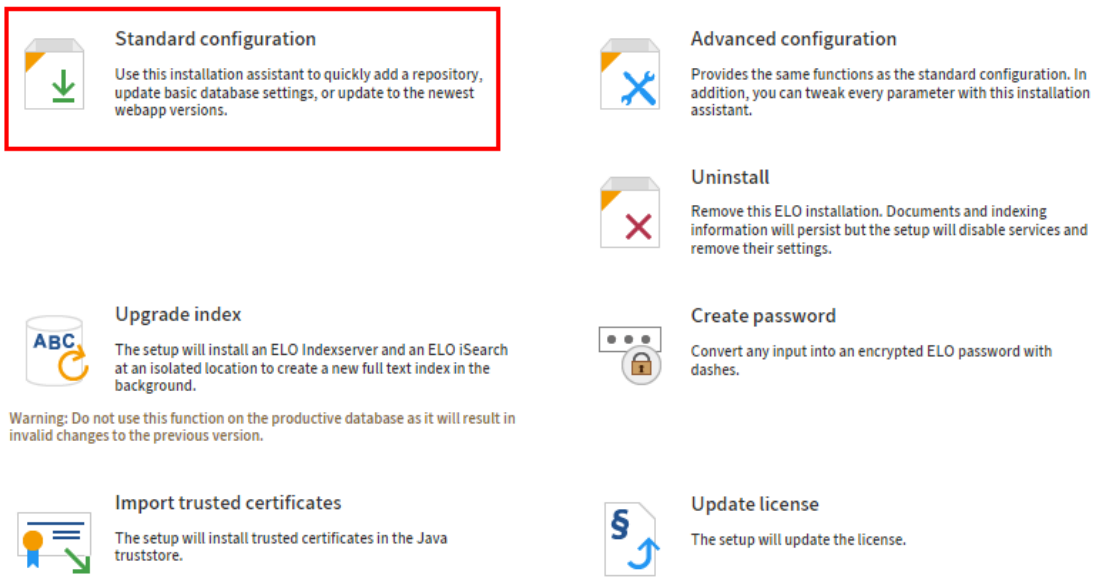
2. Enter the main password.

3. Click *Continue to ELO Server Setup*.

   The ELO Server Setup encrypts the setup configuration files (*.elosetup.conf* and *elosetup.conf*).

Information

To learn how to decrypt the files, refer to the documentation Decrypting the setup configuration.



Standard configuration

Use this installation assistant to quickly add a repository, update basic database settings, or update to the newest webapp versions.

Advanced configuration

Provides the same functions as the standard configuration. In addition, you can tweak every parameter with this installation assistant.

Uninstall

Remove this ELO installation. Documents and indexing information will persist but the setup will disable services and remove their settings.

Upgrade index

The setup will install an ELO Indexserver and an ELO iSearch at an isolated location to create a new full text index in the background.

Warning: Do not use this function on the productive database as it will result in invalid changes to the previous version.

Create password

Convert any input into an encrypted ELO password with dashes.

Import trusted certificates

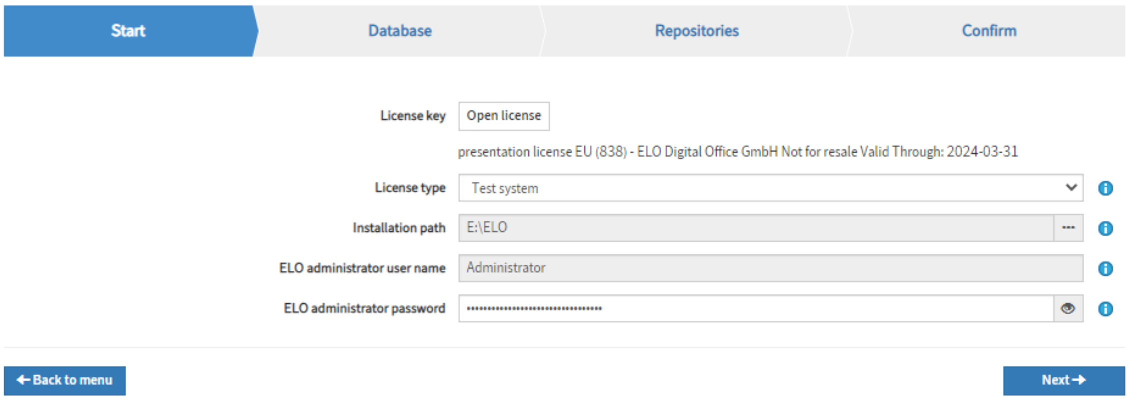
The setup will install trusted certificates in the Java truststore.

Update license

The setup will update the license.

4. Select *Standard configuration* from the menu.

   Alternatively: To change the settings during the update and add or remove modules, click *Advanced configuration*.



License key: Open license

presentation license EU (838) - ELO Digital Office GmbH Not for resale Valid Through: 2024-03-31

License type: Test system

Installation path: E:\ELO

ELO administrator user name: Administrator

ELO administrator password: ••••••••••••••••••••••••••••••••

The update menu opens. The main update options are located in the center of the window.

License key: Click *Open license* and select a valid license file.

License type: In the *License type* drop-down menu, select the type of system you want to install (production system, test system, or development system).

Installation path: The setup automatically recognizes the previously used installation path.

ELO administrator user name: The setup automatically recognizes the name of the previously used administrative user.

> **Please note**
>
> In earlier versions of ELO, a different name was assigned for the ELO administrator according to the respective language.

ELO administrator password: The password from the previous installation is automatically applied.

> **Information**
>
> Earlier versions of ELO used the same password for the administrator and the ELO Service user during installation. Since ELO 10, you can create separate passwords for these users.

5. Click *Next* to proceed with the rest of the installation process. Check the settings on the *Database*, *Repositories*, and *Confirm* tabs.

> **Please note**
>
> If you are upgrading an ELO server that uses an Oracle database, the authentication data for the database administrator is not entered to the fields automatically. You must enter the database administrator name and password to upgrade correctly.

6. To start the update process, click *Install* on the *Confirm* tab.

Update with a Postgres database

When updating from ELO 11 to higher versions in combination with the a PostgreSQL database, coding issues may occur.

Make sure that you select UTF-8 encoding.

Manual update

The method described below shows you how to update individual applications manually.

> **Information**
>
> You need to stop the ELO server if you want to perform an update in this way.

1. Download the new ELO application from the ELO SupportWeb. See the documentation provided for any dependencies between the application and other ELO server applications or client programs. Also check whether the configuration files for the application need to be changed.
2. Navigate to the `webapps` directory:

```
<ELO>/prog/webapps/
```

> **Please note**
>
> If you are running ELOenterprise, make sure you are updating the server application on the correct computer.

Check whether the application you want to update already exists in this directory. Each ELO application uses a special abbreviation (e. g. `web.war` for the ELO Web Client, `ix.war` for the ELO Indexserver, and so on).

3. Extract the updated ELO application to your hard drive and rename it (if necessary) so it has the same name as the old ELO application.

4. Stop the ELO Application Server.

5. Delete the directory for the web application in the `servers/<server name>/webapps` directory. Example:

   ```
   <ELO>/servers/ELO-SRV16-1/webapps/wf-Repository/
   ```

6. Copy the updated ELO server application to the `prog/webapps` directory so that it overwrites the old application.

7. Restart the ELO Application Server.

8. Check the status page of the updated application in your browser in the ELO Server Manager.

   If the application status is *Running* and no errors are displayed, the application has updated successfully.

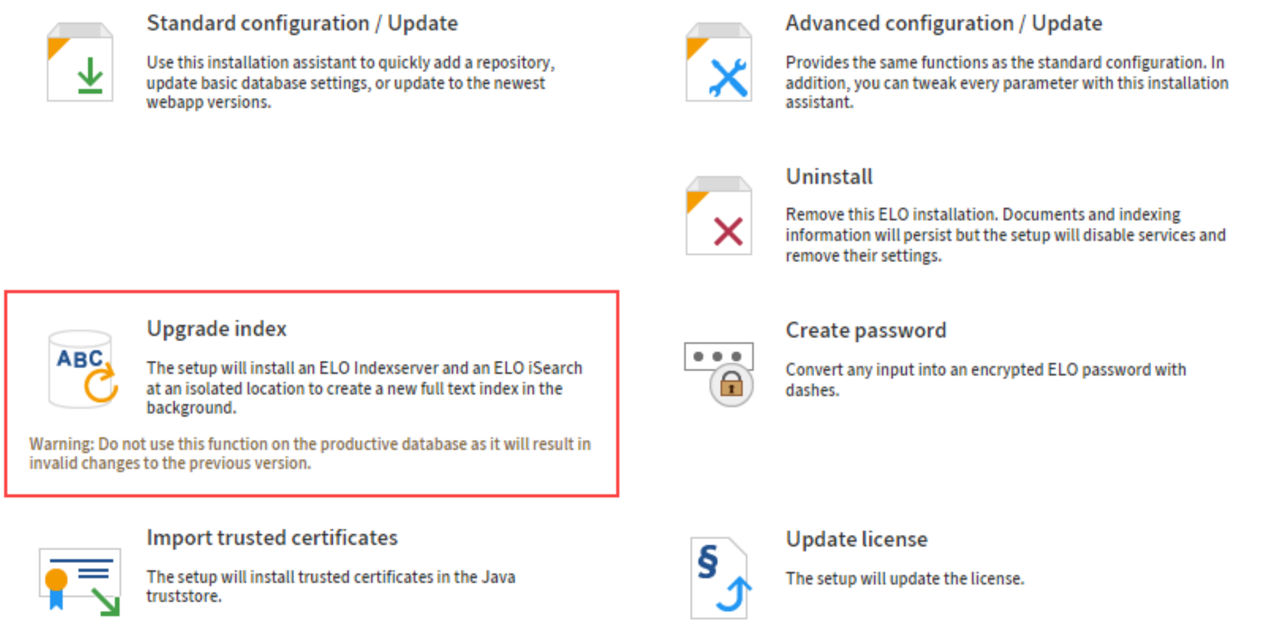
Upgrade

Upgrade index

If you update your ELO iSearch version, you may need to rebuild the full text index, such as when upgrading to a higher version.

In most live ELO installations, the amount of documents in the repository is relatively small, which means that it only takes a few hours to rebuild the index and you will have less downtime. If this applies to your repository, you can skip this section and upgrade all ELO server components at the same time.

In repositories with a large number of documents (more than 500,000), ELO provides the option to rebuild the ELO iSearch index asynchronously on a dedicated server. For more information on this topic, refer to the documentation ELO iSearch > Background reindex.



Standard configuration / Update

Use this installation assistant to quickly add a repository, update basic database settings, or update to the newest webapp versions.

Advanced configuration / Update

Provides the same functions as the standard configuration. In addition, you can tweak every parameter with this installation assistant.

Uninstall

Remove this ELO installation. Documents and indexing information will persist but the setup will disable services and remove their settings.

Upgrade index

The setup will install an ELO Indexserver and an ELO iSearch at an isolated location to create a new full text index in the background.

Warning: Do not use this function on the productive database as it will result in invalid changes to the previous version.

Create password

Convert any input into an encrypted ELO password with dashes.

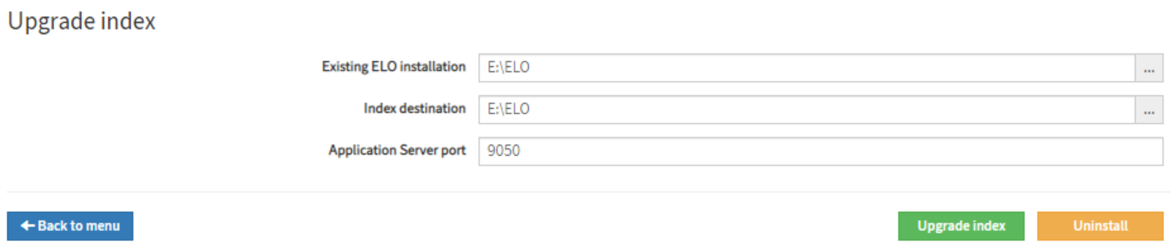
Import trusted certificates

The setup will install trusted certificates in the Java truststore.

Update license

The setup will update the license.

1. To start the upgrade process, click *Upgrade index* in the ELO Server Setup.



Upgrade index

Existing ELO installation	E:\ELO
Index destination	E:\ELO
Application Server port	9050

The *Upgrade index* screen opens.

Once you have entered all options and then clicked *Upgrade index*, a minimal ELO server with ELO Indexserver and ELO iSearch will be installed to the computer.

Existing ELO installation: Enter the path to your existing ELO server installation here.

Index destination: Path where you want the rebuilt index to be saved. Once you have entered an *Existing ELO installation*, the entry is copied into this field. However, it is recommended to store the index on an SSD or faster medium.

Application Server Port: Choose a free port that will be used by the new Apache Tomcat. Do not use the same port and computer that you use for your ELO server.

Click *Upgrade index* to install the ELO server and client components. The screen switches to a status display.

When the new search index has been created, click *Uninstall* to remove the new Apache Tomcat server and its associated Windows services and files from the system. This will not delete the search index.

ELO Transport

Basics

ELO Transport is used to transfer data from one ELO repository to another. This includes data such as font colors, users, workflow templates, keyword lists, metadata forms, folders, and documents.

ELO Transport is based on ELO Automation Services (ELOas). You can transfer data entirely using scripts or ELOas. Alternatively, you can create a transport file with the ELO Administration Console (see below) and import it into the target repository with an ELOas ruleset.

Requirements

Since a few additions have been made in ELOas to enable the ELO Transport module, version 8.00.014 of ELOas and version 8.00.045 of the JavaScript libraries are required.

The transport system itself includes three JavaScript libraries:

- tfer: Indexserver JSON objects
- tfex: Transport export
- tfim: Transport import

These libraries provide the functions to export and import zipped JSON objects.

Function calls at a glance

The export is performed by the `doExport` function in the `tfex` library. This is called in a normal ELOas ruleset. The function is called with a string containing a JSON object of the areas to be exported and a file name for the ZIP folder to be created.

```
<script>
  var desc = String(Sord.desc);
  tfex.doExport(desc, "\\\\testmachine\\temp\\json.zip");
</script>
```

If the transport is created using the ELO Administration Console, the export process should be started as follows:

```
<script>
  var desc = String(ix.downloadAsString(Sord));
  tfex.doExport(desc, "\\\\testmachine\\temp\\json.zip");
</script>
```

The areas to be exported can be defined directly as a string in the ruleset. However, it is more flexible if they are saved in a separate document or folder (for example, in the extra text area of the metadata). The ruleset is then triggered for the object, reads the extra text, and runs the export function. It usually makes the most sense to place the export process in a triggered ruleset (`<interval>0M</interval>`).

The import is triggered by the `checkForImport` function in the `tfim` library. The call checks whether a transport file is available, then processes it if it is found. Only the name of the expected transport file is transferred as parameter. After processing, the file is renamed or deleted. This ruleset call can occur in fixed intervals, since if a transport file is missing, no actions can be initiated. However, it can also be triggered manually.

```
<script>
  tfim.checkForImport(""\\\\testmachine\\temp\\json.zip");
</script>
```

Architecture

Upon export, a transport file is generated that contains all objects from the transport definition. This transport file is a normal ZIP file, and each object is its own file object within the ZIP data stream. Within the transport file, all objects are identified by their GUID. The only exception to this is the markers, which do not possess their own GUID. Therefore, they can only be transferred 1:1 in a block.

Transferring the file from the export source to the import directory is not part of the standard product. It must be implemented within each project according to the requirements of the situation. In the simplest case, however, export and import can use a common transfer directory.

If the copying is performed manually or with a separate program, please note that the generation of the import file must be an atomic process. This means that the import file (*json.zip* in the example) may at no point exist only partially. In such cases, the worst-case scenario would be that the import process starts with an incomplete file, then cancels or generates an incomplete transport. The simplest way to fulfill this requirement is to copy the file to the target directory under a different name, then rename it to the correct one. The export works in the same way, which makes it possible to configure the export and import directly to the same directory.

Upon import, the transport file is read out in the same order in which it was written. Therefore, if the ZIP folder is edited by other programs, care should be taken to retain the same file order. Otherwise it may lead to incomplete or erroneous imports. Each partial file in the ZIP folder contains a JSON object with data about the entry. Only documents have an additional stream with the file contents. Upon import, first an attempt is made to read in an object based on its GUID. If it already exists, it is updated using the transport data. If it does not exist, it is created.

Folders and documents play a special role in this. These objects do not only have their own GUID, but rather also contain a parent GUID that defines their filing location. If the parent does not exist, the object cannot be created and the transport will cancel. However, the parent GUID is only required and checked for new objects. If an object already exists, the parent GUID will be ignored. Objects will also not be moved within the target repository after creation. This makes it possible for partial trees in the source repository to be structured differently from the target repository. Reassignments in the target repository will not be reversed by subsequent transports.

Export

The export process expects a JSON object and a description of the objects to be transported. You can currently transfer the following objects:

- Font colors
- Users
- Workflow templates
- Keyword lists
- Forms
- Folders
- Documents

The objects to be exported must be added to the description in the order shown above. Otherwise, for example, a metadata form could be imported for which no corresponding workflow template exists. If objects are not added in this order, the administrator must ensure that all dependent objects exist.

```
[
{"type": "marker", "filter": ""},
{"type": "user", "guid": "(3B8E73FC-F743-D0DF-3C6E-BEFE9620CDCF)"},
{"type": "user", "guid": "(809543F8-3659-5199-A81F-41AA330A9EF8)"},
{"type": "wftemplate", "guid": "(0B5E6CDB-129D-E9DD-64D6-F334DAF361A5)"},
{"type": "keywords", "guid": "ABCDA"},
{"type": "mask", "guid": "27"},
{"type": "mask", "guid": "8"},
{"type": "mask", "guid": "(52952663-E6DB-4D49-BBCE-F5049D4128F4)"},
{"type": "sord", "guid": "(278AEB0A-CB83-95EA-F064-F5790BDF48FB)"},
{"type": "sord", "guid": "(FF0D4E6F-4BDD-AA35-C6AC-1798E677A830)",
        "createPath": "ARCPATH:¶Local folder¶Transfer"}
]
```

(tbc.)

With the ELO Administration Console

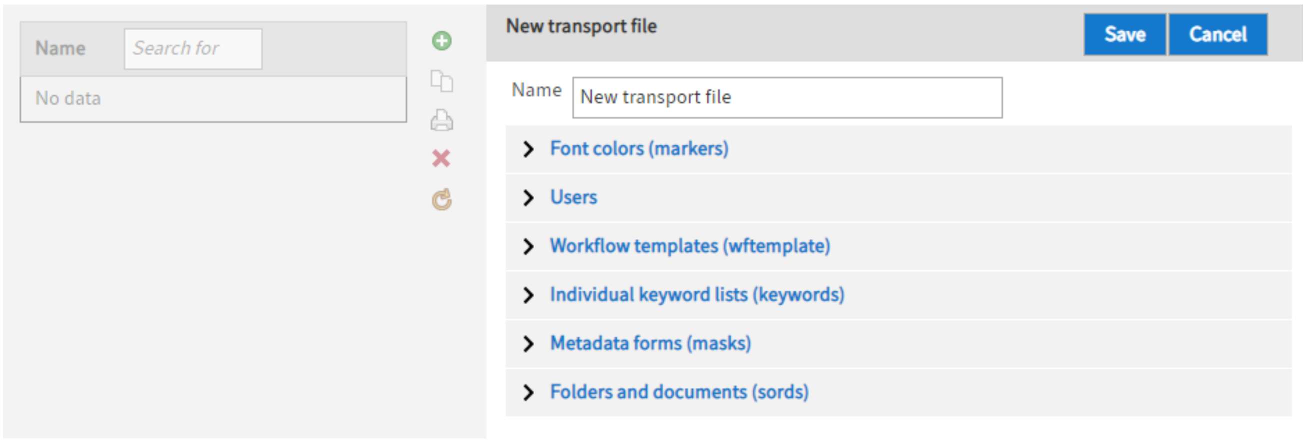
You can create a new transport file in the *ELO Transport* menu in the ELO Administration Console. The information is stored in a file that can be imported into an ELO repository.

The ELO Transport configuration is filed to the *// ELOas Base // TransferConfig* folder as a JSON file.

In this case, the JSON configuration can be read as follows and used in the ELOas rule for export:

```
"var desc = String(ix.downloadAsString(Sord));"
```

Create transport file



Information

To view a drop-down list in an input field, you need to move the cursor over the input field and press the space key.

New transport file: Click the green button with the white plus sign to create a new transport file.

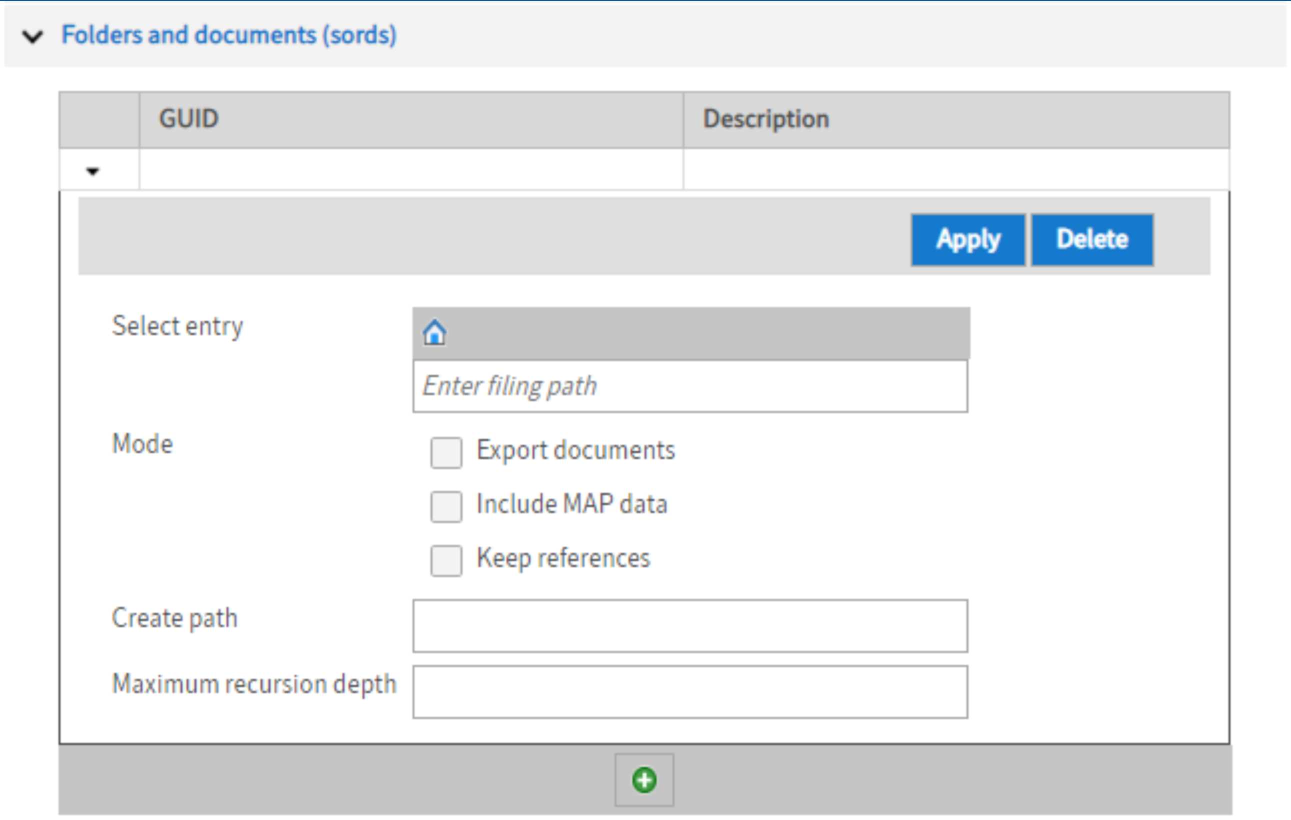
Font colors (markers): Font colors can be transferred along with the transport file.

Users: Enter the user and group to be transferred here.

Individual keyword lists (keywords): Enter the keyword lists to be transferred here.

Metadata forms (masks): Enter the metadata forms to be transferred here.

Folders and documents (sords): Select the folders and documents to be transferred here.



Apply: Select the folders and documents that you want to add to the transport set.

Information

To view a drop-down list in an input field, you need to move the cursor over the input field and press the space key.

Delete: Click this button to remove the dialog box and the selected folders from the current view.

Export documents: The selected documents are exported as well.

Include MAP data: The advanced fields (map fields) are included in the metadata when creating an ELO Transport set.

Keep references: If the *Keep references* option is enabled, references are transferred as well.

CreatePath: Enter the path in the target repository here. The syntax of the path must use pilcrow signs. Example: *¶Repository¶Invoices*.

Maximum recursion depth: Specify the number of repository levels to be included in the ELO Transport set.

Examples

(tbd.)

Migration of a Microsoft SQL database to PostgreSQL

Introduction

In some cases, you may be required to migrate to another database technology.

This documentation describes how to use the tool MS2PGSQL, which can perform migrations from Microsoft SQL to PostgreSQL in the ELO environment.

You will find the tool in the ELO SupportWeb under *Modules > ELO Indexserver (ELOix) > Datenbankmigration MS to Postgres 1.0.0 .10*.

The following topics are covered:

- Installation
- Migration
- Error control

Installation

The program does not require installation since it is provided as a module with its own Java version and can be started from the batch file *MS2PGSQL.bat*. The batch file only ensures that the program is started with the local Java version.

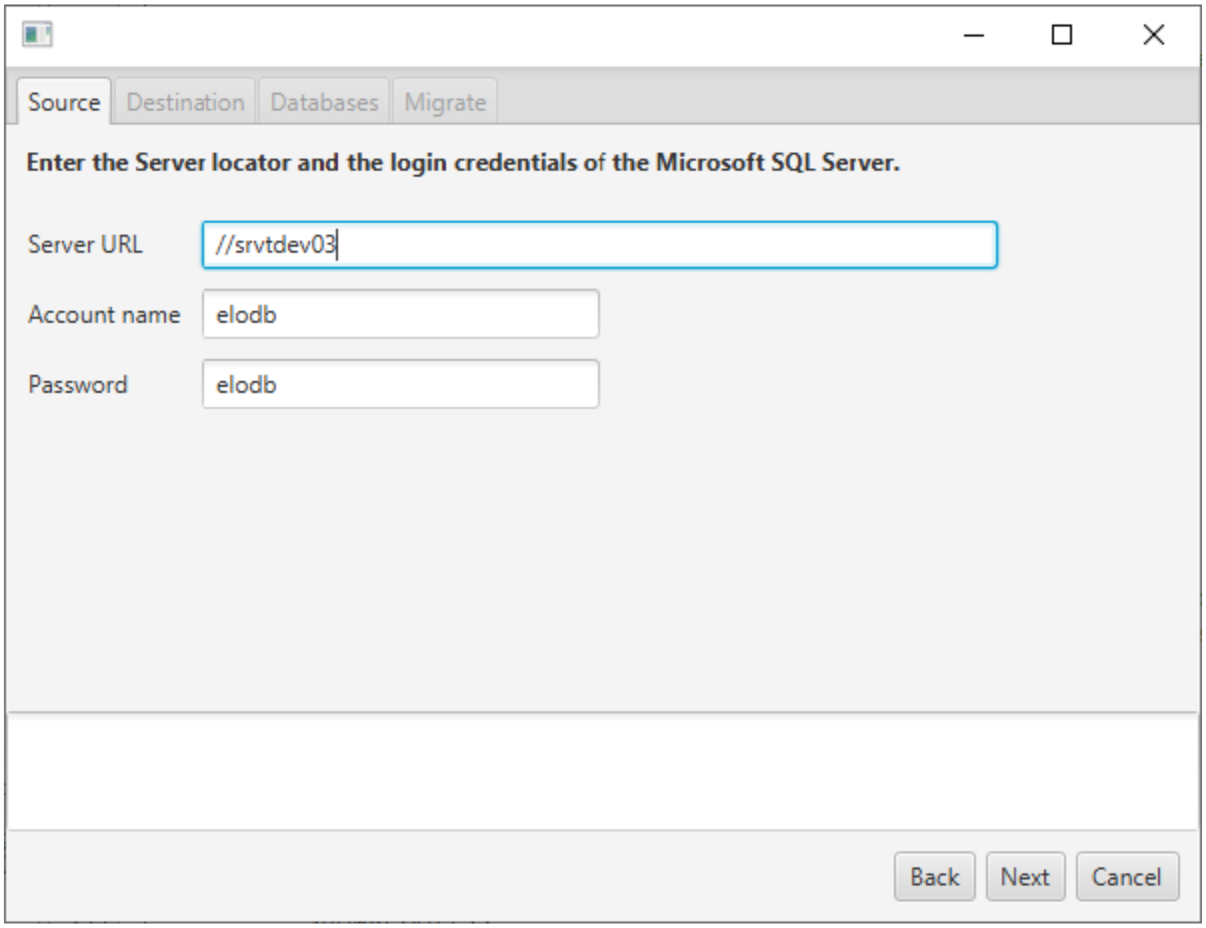
```
.\bin\javaw.exe com.elo.ms2pgsql.App
```

You will achieve the best performance by running it on the PostgreSQL server. However, you can run it on any client computer.

The older version of the migration program used a SQL script file, which contained the structure of the database. The appropriate script had to be selected for each ELO version. This is no longer necessary or possible in the current version. Since additional tables are created dynamically for the aspects from version ELO 21, you can no longer work with a static database description. Instead, the migration program reads the structure of the source database and uses this information to create the target database.

Migration

After starting, the Microsoft SQL Server is queried with the source database.

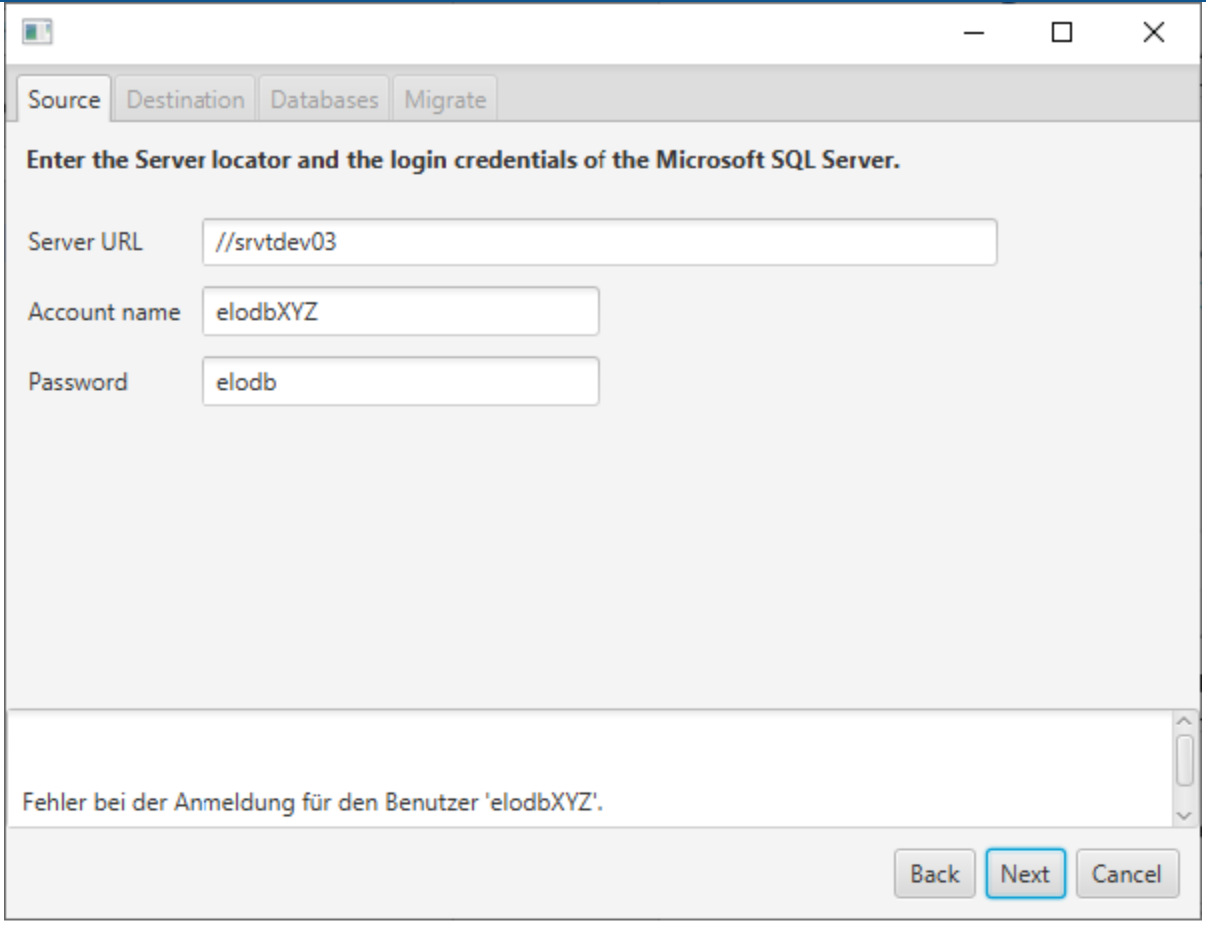


Enter the Server locator and the login credentials of the Microsoft SQL Server.

Server URL: //srvtdev03
Account name: elodb
Password: elodb

You need to enter the server URL and account that has full access to the source data.

When you click *Next*, the database connection is tested.



Enter the Server locator and the login credentials of the Microsoft SQL Server.

Server URL: //srvtdev03

Account name: elodbXYZ

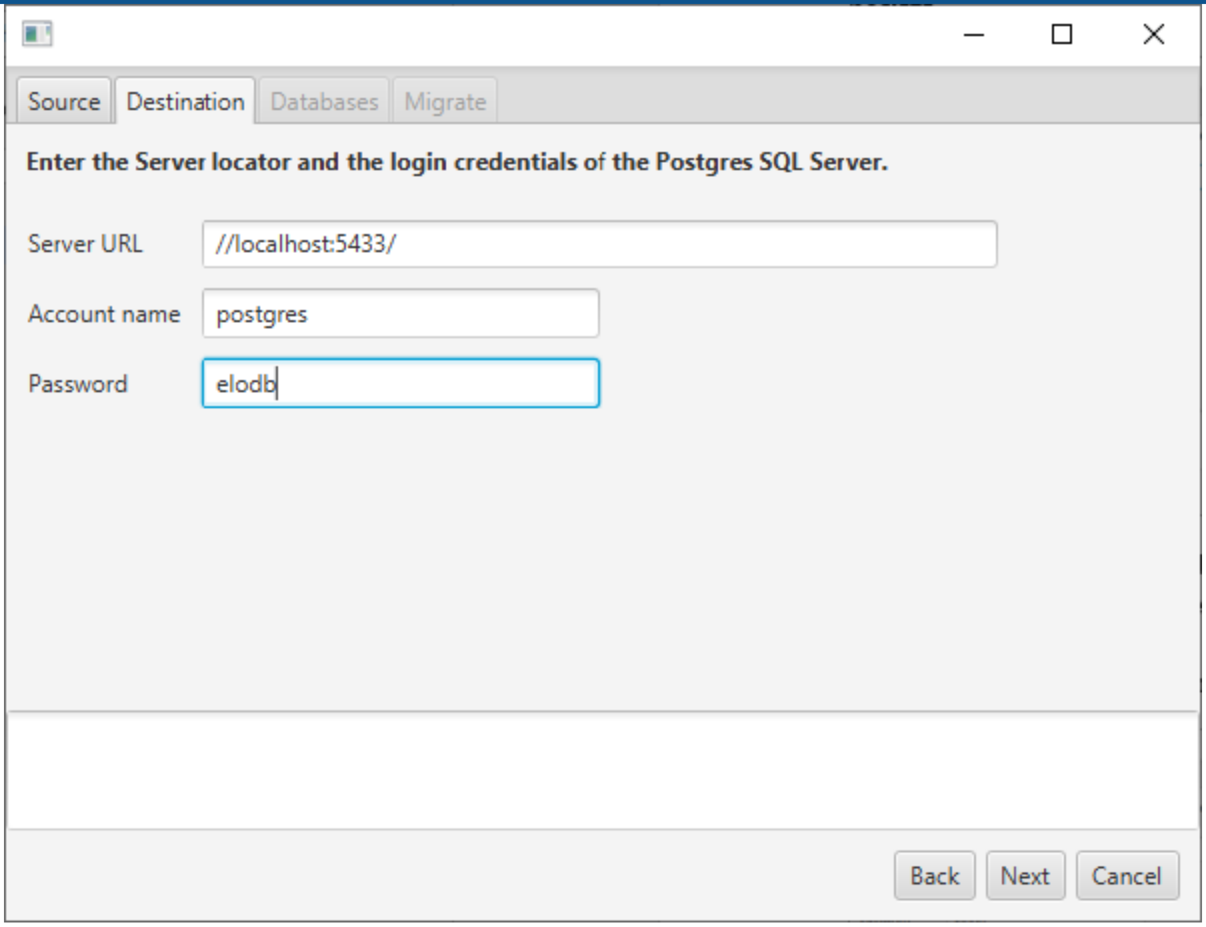
Password: elodb

Fehler bei der Anmeldung für den Benutzer 'elodbXYZ'.

If you can't access the server, the program stops on the *Source* tab and an error message is displayed.

If you entered an invalid URL, a longer timeout from the network layer can delay the error message.

If the connection is established, the program proceeds to the *Destination* tab.



Enter the Server locator and the login credentials of the Postgres SQL Server.

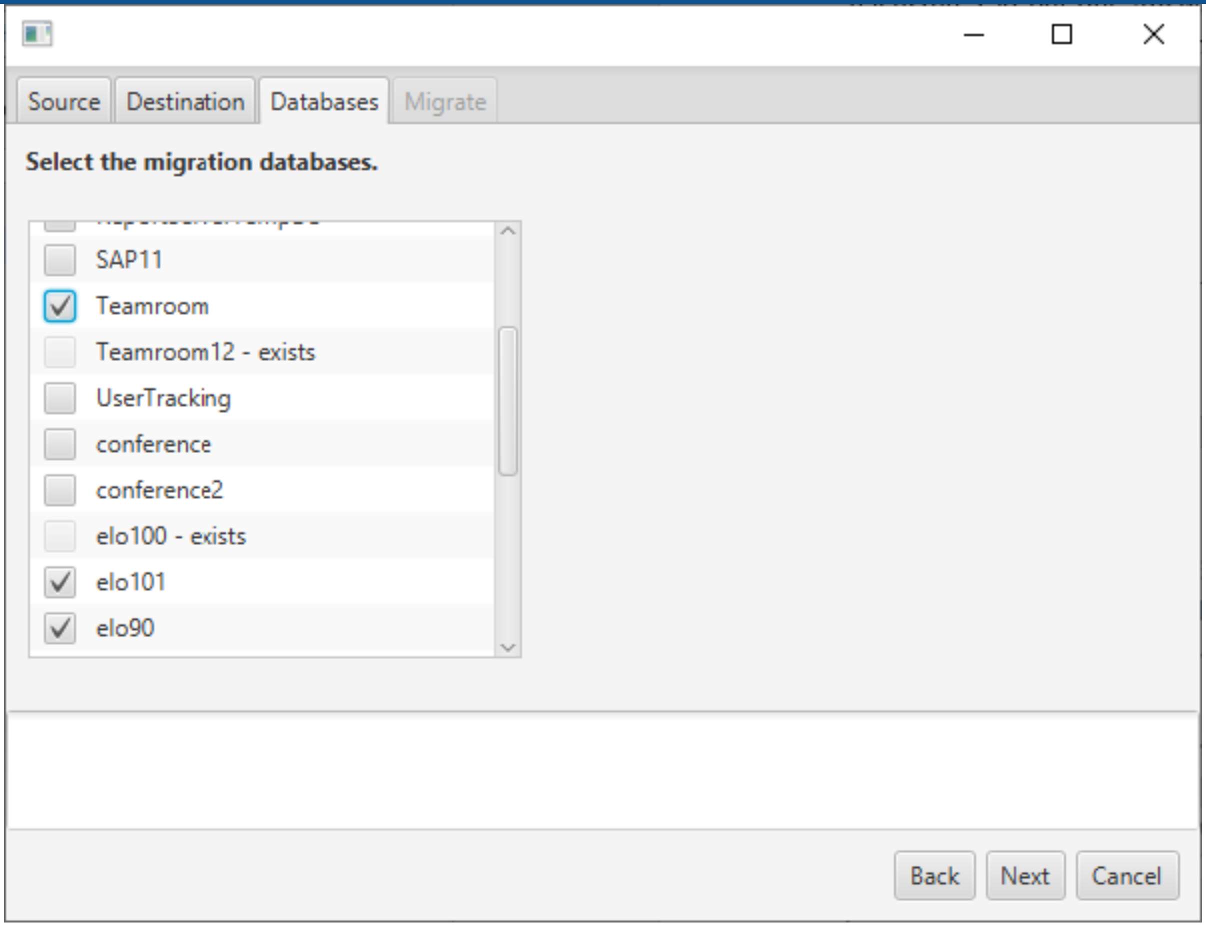
Server URL: //localhost:5433/
Account name: postgres
Password: elodb

This is where you enter the parameters for the target database server.

Please note

Always end the target URL with the character '/'. Also, it must not contain the name of the database. These are only specified on the next tab when selecting the databases. The selected user must have sufficient rights to create databases.

When you click *Next*, the connection is tested again. The program only proceeds to the *Databases* tab if a connection can be established.



Select the migration databases.

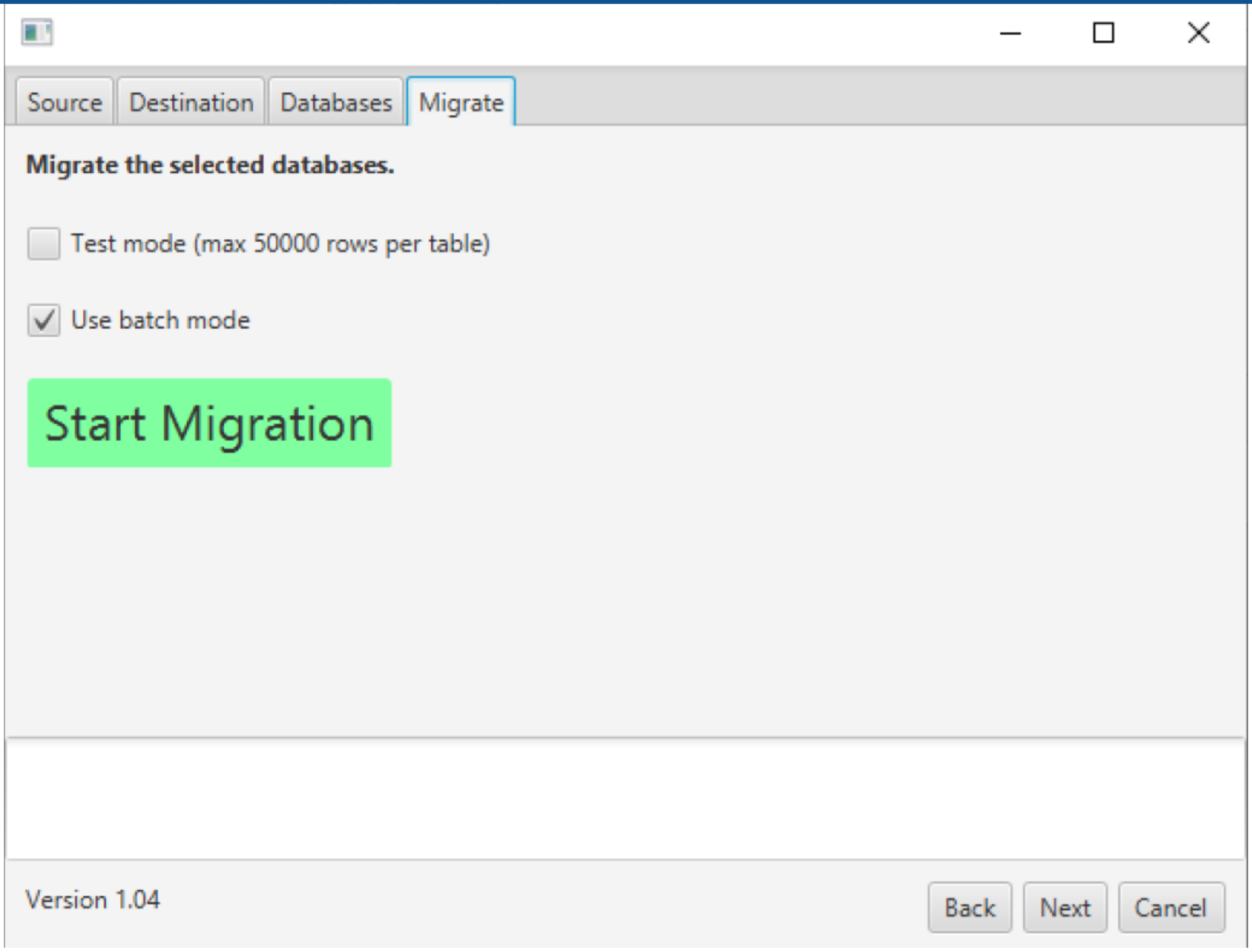
- [ ] SAP11
- [x] Teamroom
- Teamroom12 - exists
- [ ] UserTracking
- [ ] conference
- [ ] conference2
- elo100 - exists
- [x] elo101
- [x] elo90

On the *Databases* tab, you can select the databases that you want to migrate. You can only selected database names that do not exist on the target server. Hybrid databases are not possible.

Information

If the migration is terminated or needs to be repeated for some other reason, you first need to delete the existing database. However, you don't need to abort the migration program. In this case, you can go back to the previous tab, delete the database, and then proceed to the *Databases* tab again. The databases are read again and checked.

Click *Next* to proceed to the *Migrate* tab.


Migrate the selected databases.

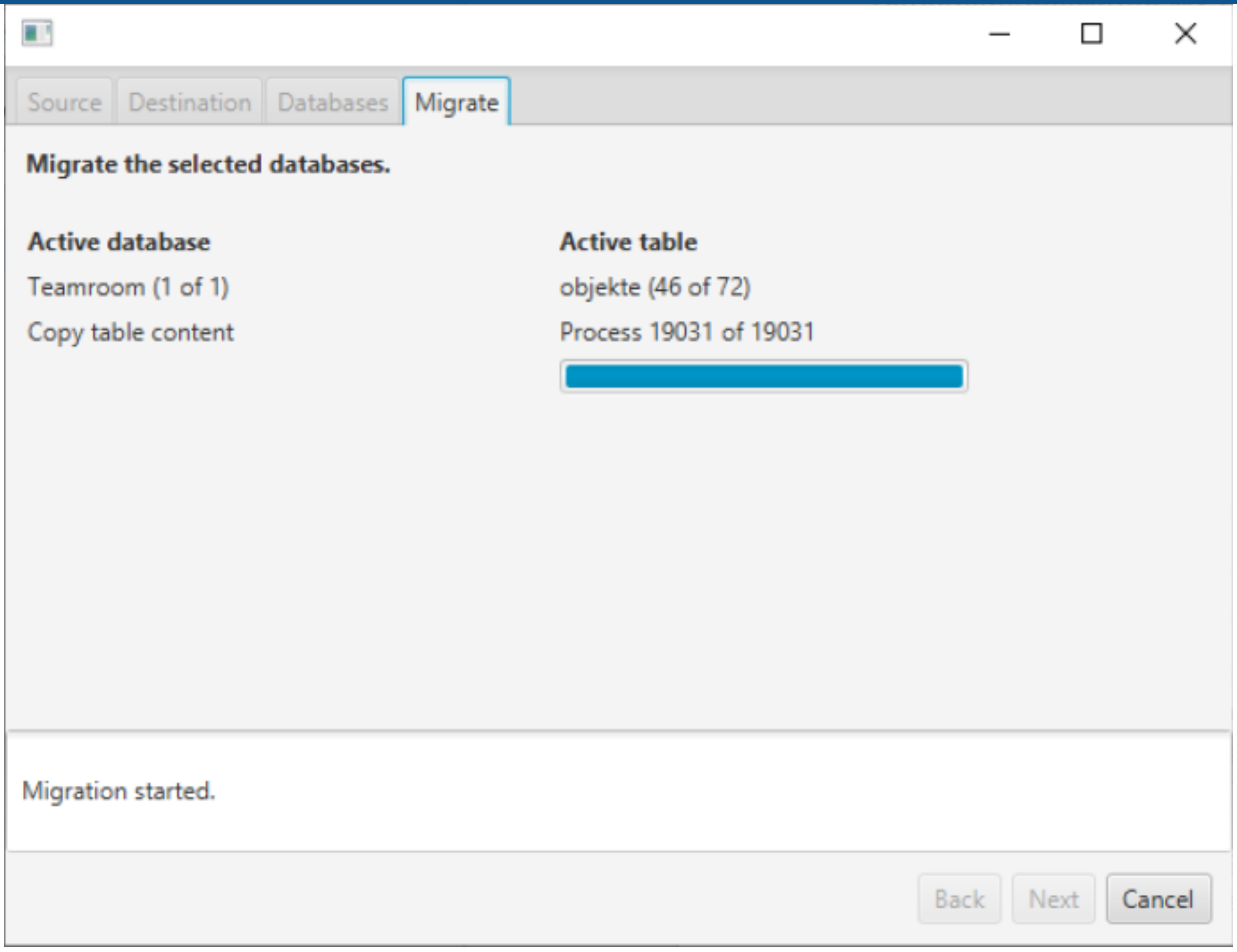
☐ Test mode (max 50000 rows per table)

✓ Use batch mode

You can set the following options:

- Test mode: In this mode, the database is created and all tables are copied. However, a maximum of 50,000 rows are copied from each table. This allows you to do a quick test to pinpoint fundamental issues. An estimate of the time required to complete the migration is indicated at the end.
- Use batch mode: This option transfers the data to the PostgreSQL database in batch mode. It is about 10 times faster but has the disadvantage that if an error occurs, it is hard to tell which entry is causing the problem. In this case, you can disable the batch mode and look at the report file to find out where the error has occurred.

Click *Start Migration* to start the migration.



Migrate the selected databases.

Active database

Teamroom (1 of 1)

Copy table content

Active table

objekte (46 of 72)

Process 19031 of 19031

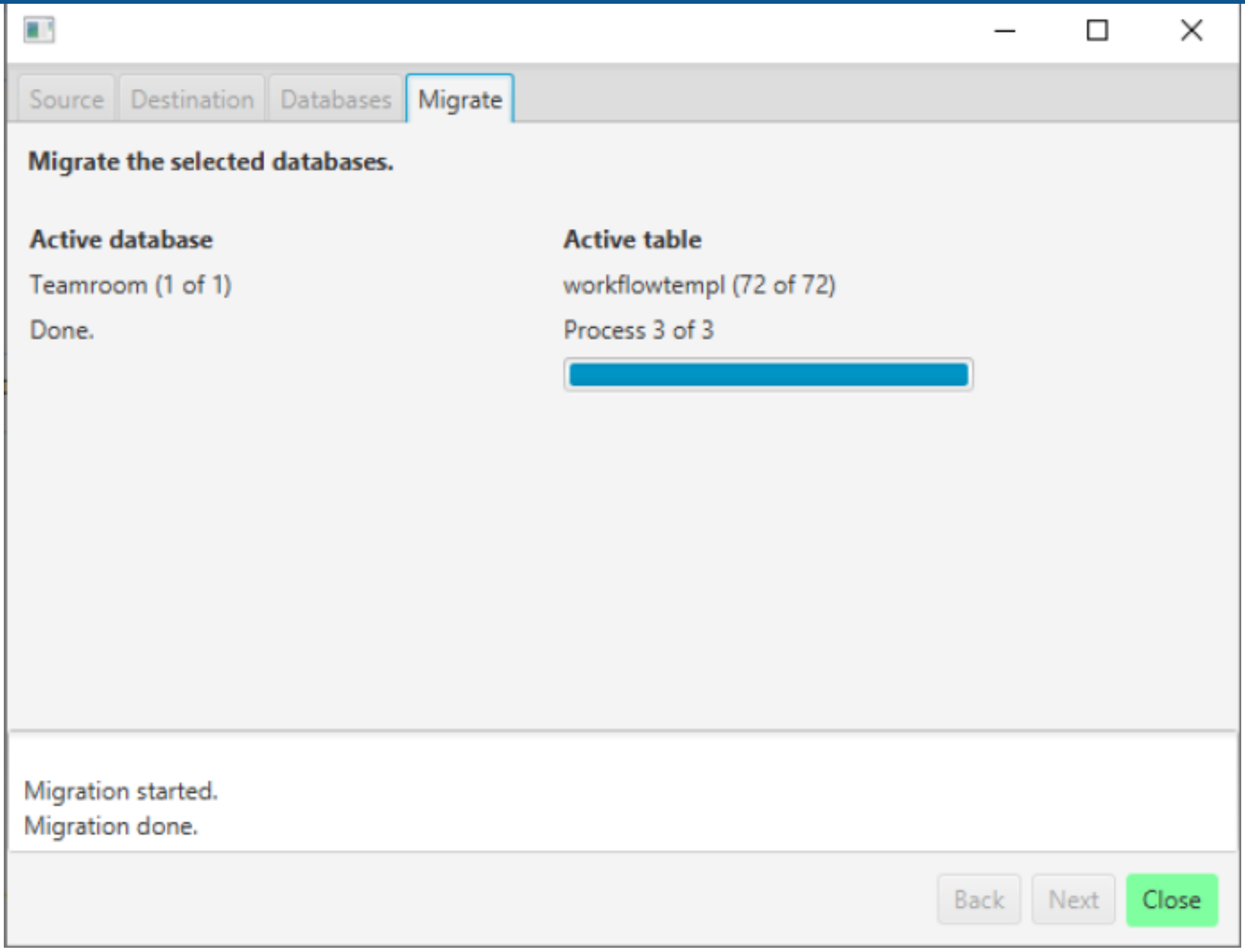
Migration started.

During migration, the metadata of the source database is read first. Next, the target database is created, the tables within the database are generated, and the data is copied. Finally, the sequences of the source database are read and replicated in the target database. All steps are shown in the status display and a bar indicates the progress of the copy operation.

Please note

With very large tables, the check at the end of the copy operation can take a long time to complete.

For unknown reasons, the `COUNT(*)` operation under PostgreSQL is very expensive. In addition, the `SELECT` operation is expensive for tables with a timestamp column, because this `SELECT` needs an `ORDER BY <timestamp column>` so that the entries can be entered on the target page in the correct order. If the progress bar remains at these points for a long time, it is not a sign that the program is not responding.



Migrate the selected databases.

Active database

Teamroom (1 of 1)

Done.

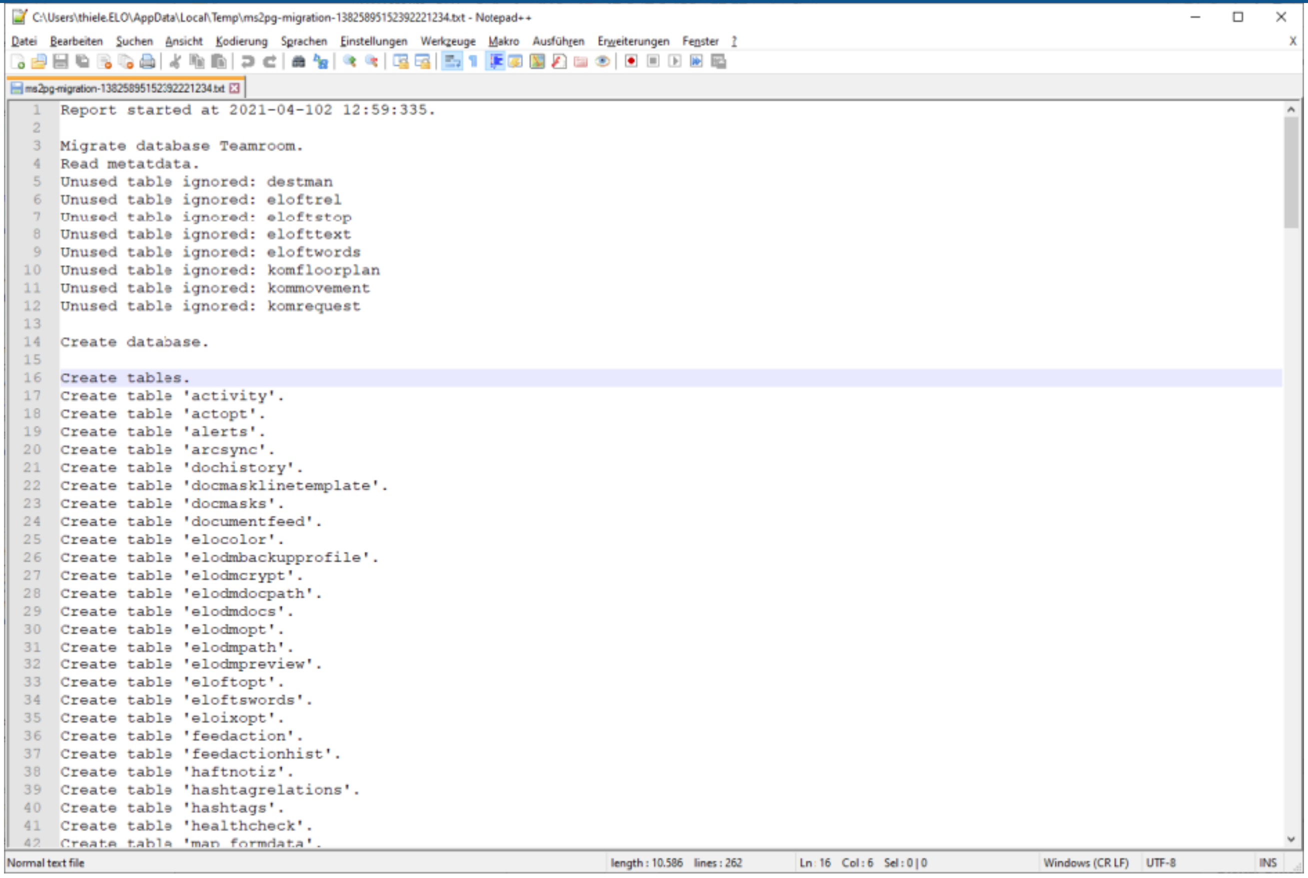
Active table

workflowtempl (72 of 72)

Process 3 of 3

```
Migration started.
Migration done.
```

If migration is successful, the report is automatically displayed and the *Cancel* button is renamed to *Close* and displayed in green.


C:\Users\thiele.ELO\AppData\Local\Temp\ms2pg-migration-13825895152392221234.txt - Notepad++
Datei Bearbeiten Suchen Ansicht Kodierung Sprachen Einstellungen Werkzeuge Makro Ausführen Erweiterungen Fenster ?
ms2pg-migration-13825895152392221234.txt
Normal text file
length : 10.586 lines : 262
Ln : 16 Col : 6 Sel : 0 | 0
Windows (CR LF) UTF-8 INS

```
Report started at 2021-04-102 12:59:335.

Migrate database Teamroom.
Read metatdata.
Unused table ignored: destman
Unused table ignored: eloftrel
Unused table ignored: eloftstop
Unused table ignored: elofttext
Unused table ignored: eloftwords
Unused table ignored: komfloorplan
Unused table ignored: kommovement
Unused table ignored: komrequest

Create database.

Create tables.
Create table 'activity'.
Create table 'actopt'.
Create table 'alerts'.
Create table 'arcsync'.
Create table 'dochistory'.
Create table 'docmasklinetemplate'.
Create table 'docmasks'.
Create table 'documentfeed'.
Create table 'elocolor'.
Create table 'elodmbackupprofile'.
Create table 'elodmcrypt'.
Create table 'elodmdocpath'.
Create table 'elodmdocs'.
Create table 'elodmopt'.
Create table 'elodmpath'.
Create table 'elodmpreview'.
Create table 'eloftopt'.
Create table 'eloftswords'.
Create table 'eloixopt'.
Create table 'feedaction'.
Create table 'feedactionhist'.
Create table 'haftnotiz'.
Create table 'hashtagrelations'.
Create table 'hashtags'.
Create table 'healthcheck'.
Create table 'map_formdata'.
```

The report file can be saved for documentation purposes. It contains all the important steps of the migration. You can see which tables were created and copied, how many data sets they each contained, and which sequences were created.

If several databases were migrated simultaneously, the report contains a separate section for each database along with information about whether any errors occurred:

```
MIGRATION OF DATABASE TEAMROOM DONE, NO ERRORS REPORTED.
```

Finally, there is a text at the end that indicates whether all databases were copied without errors or whether there were problems in a database.

```
Migration done, no errors reported.
```

Error control

When doing a migration, there is always a risk that a problem or incompatibility will occur at some point that prevents you from migrating the database. Several checks were integrated into the migration program to minimize this risk. All errors that are detected are listed in the report. The administrator must decide how to fix the problems (e.g. missing permissions, insufficient memory), whether the problems were only temporary (network malfunction), or whether they can be fixed manually at the SQL level.

The migration program lists in the first section of the report which tables were excluded from the migration. In the case of older repositories, these are tables that existed in previous ELO versions but are no longer used in current ELO versions (e.g. the old com* tables):

```
Report started at 2021-04-102 12:59:335.

Migrate database Teamroom.
Read metatdata.
Unused table ignored: destman Unused table ignored:
        eloftrel Unused table ignored: eloftstop Unused table ignored:
        elofttext Unused table ignored: eloftwords Unused table ignored:
        komfloorplan Unused table ignored:
        kommovement Unused table ignored: komrequest
```

Next, the report lists that the database was created and which tables were created:

```
Create database.

Create tables.
Create table 'activity'.
Create table 'actopt'.
Create table 'alerts'.
Create table 'arcsync'.
Create table 'dochistory'.
Create table 'docmasklinetemplate'. …
Create table 'workflowhistory'. Create table 'workflowtempl'.
```

If not all tables were visible to the migration program due to permission restrictions, the program does detect this automatically. If there is any doubt, the administrator must manually compare the number of tables.

In the next step, the table contents are copied. The migration program lists how many entries were copied for each table:

```
Copy database content.
Migrate table 'activity'.
Migration check ok at activity -  rows expected: 0, found: 0
Migrate table 'actopt'.
Migration check ok at actopt -  rows expected: 15, found: 15
Migrate table 'alerts'.
Migration check ok at alerts -  rows expected: 0, found: 0
Migrate table 'arcsync'.
Migration check ok at arcsync -  rows expected: 0, found: 0
Migrate table 'dochistory'.
Migration check ok at dochistory -  rows expected: 14859, found: 14859
```

The number of rows in the source database, the number of rows read, and the number of rows in the target database are checked for each table. Inconsistencies are listed as errors in the report. There is also an error message if an error occurs while writing the data:

```
Migrate table 'dochistory'.
ERROR: ### Migration error at dochistory -  rows expected: 14859,
       found: 14842, items read: 14859 …
Migrate table 'report'.
ERROR: ### Migration error at report -  rows expected: 578638,
       found: 522929, items read: 578638
```

In test mode, the following information is displayed in the case of larger tables:

```
Migrate table rows 'report'.
Test mode,
        number of rows reduced from 1410254 to 50000 Migration check ok at report -
                rows expected: 50000, found: 50000
```

The migration program indicates that the report table has 1,410,254 rows but only the first 50,000 of them were migrated. Therefore, only 50,000 rows are expected in the target table, and were also found.

At the end of the report, you will find information about the time elapsed and an estimate of the time required to complete the migration.

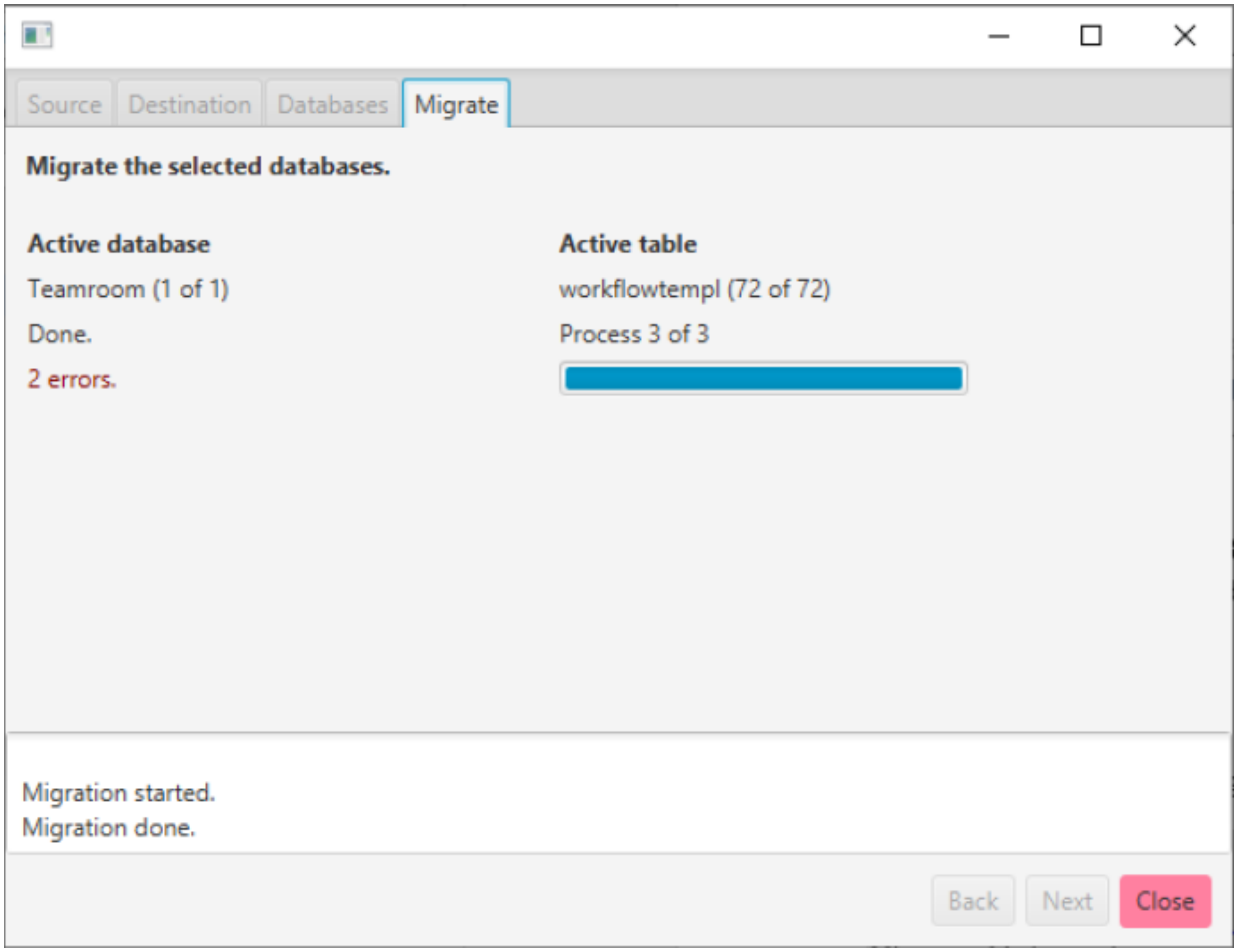
```
Migration of database ELO20 done, no errors reported.

Time used (seconds): 6

Estimated time used for full migration (seconds): 27

Migration done, no errors reported.
```

In this case, the estimated time was 27 seconds and the migration took 30 seconds to complete.



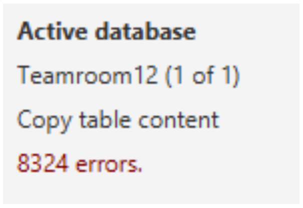
Migrate the selected databases.

Active database
Teamroom (1 of 1)
Done.
2 errors.

Active table
workflowtempl (72 of 72)
Process 3 of 3

Migration started.
Migration done.

The migration program also indicates if errors have occurred. This information is displayed while the migration is in progress, allowing the administrator to decide whether to abort or not.

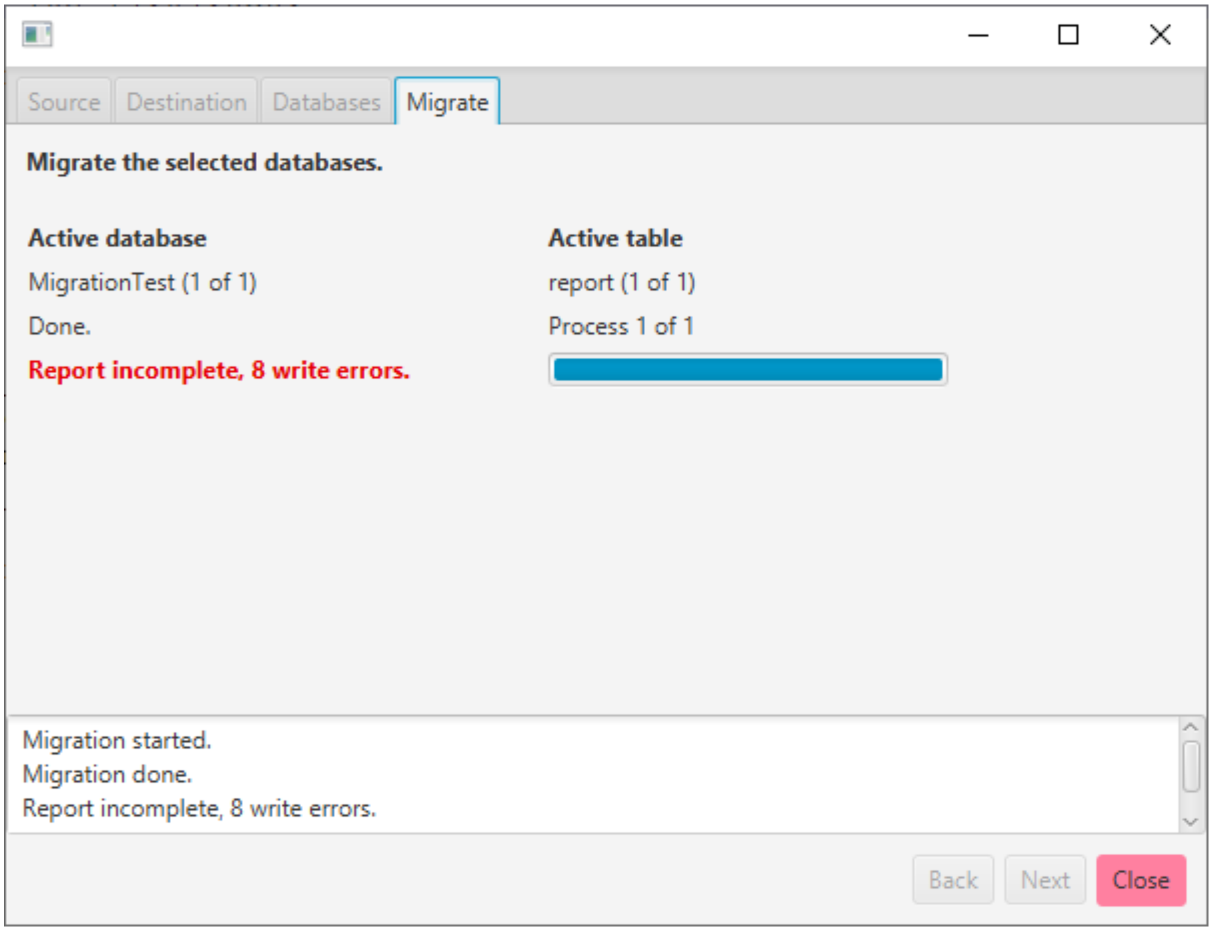


Active database
Teamroom12 (1 of 1)
Copy table content
8324 errors.

In this example, the error was artificially created by manipulating data in the target database during migration. Since no database error occurred, only a difference in the number of expected rows was detected. Database errors usually occur when saving, e.g. due to lack of memory. In this case, the error is also indicated in the user interface and the report:

```
Migrate table 'elocolor'.
ERROR: (E10E1000-E100-E100-E100-E10E10E10E20) ::
        ERROR: Value too long for type character varying(10)
ERROR: (E10E1000-E100-E100-E100-E10E10E10E21) ::
        ERROR: Value too long for type character varying(10)
ERROR: (E10E1000-E100-E100-E100-E10E10E10E22) ::
        ERROR: Value too long for type character varying(10)
ERROR: (E10E1000-E100-E100-E100-E10E10E10E23) ::
        ERROR: Value too long for type character varying(10)
ERROR: ### Migration error at elocolor - rows expected:
        4, found: 0, items read: 4
```

If errors occur while the report is being written, they cannot of course be written to the report. This can be the case if the temp directory is full, for example.



Migrate the selected databases.

Active database
MigrationTest (1 of 1)
Done.
Report incomplete, 8 write errors.

Active table
report (1 of 1)
Process 1 of 1

```
Migration started.
Migration done.
Report incomplete, 8 write errors.
```

In this case, a warning is displayed in the user interface that the report file is incomplete.

Final remarks

Even though the migration program takes many precautions to prevent or at least indicate errors during migration, the administrator is still required to conduct extensive testing to see if the data was migrated completely and the target repository is working correctly.

The migration program only copies the database tables and the sequences. Indexes are not copied because they are checked by the ELO Indexserver at startup and automatically created where necessary. If you created your own indexes manually in the source repository, you also need to create them manually in the target repository after migration. This means that the ELO Indexserver can take a long time to start after the migration since it needs to create and populate all indexes at this point.

The same applies for views. These are not copied either because they are automatically created by the ELO Indexserver. Custom views have to be created manually after the migration.

The migration program dynamically copies the database structure and could theoretically also copy third-party databases. However, we do not recommend this, as we only work with database parts and data types that are used in an ELO environment. The migration also performs a few ELO-specific conversions, e.g. `MS Timestamp` columns (which do not contain a timestamp but a kind of 64-bit transaction counter) are converted to `BIGINT` columns and any `NOT NULL` values are removed (since these columns are not frequently used by the ELO Indexserver and are therefore no longer completed).