

The Tamino 3.1 Schema and Data Migration Tool

This documentation describes the migration tool supplied with Tamino 3.1 for the schema migration from *TSD 2* to *TSD 3* and the data migration from Tamino 2.2 data format to the new data format in. It is divided in the following sections:

- Users Guide to the Tamino 3.1 Schema and Data Migration Tool
 - The Tamino 3.1 Schema and Data Migration Tool - Reference for the Command Line Version
 - The Tamino 3.1 Schema and Data Migration Tool - Command Reference for the GUI Version
 - Technical Details of the Tamino Migration Tool
-

Users Guide to the Tamino 3.1 Schema and Data Migration Tool

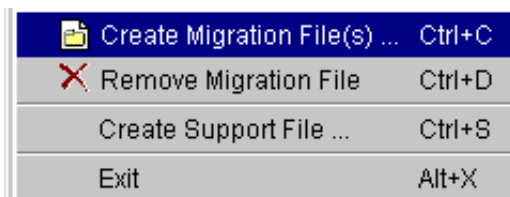
A Typical Migration Session using the Migration Tool

Caution:

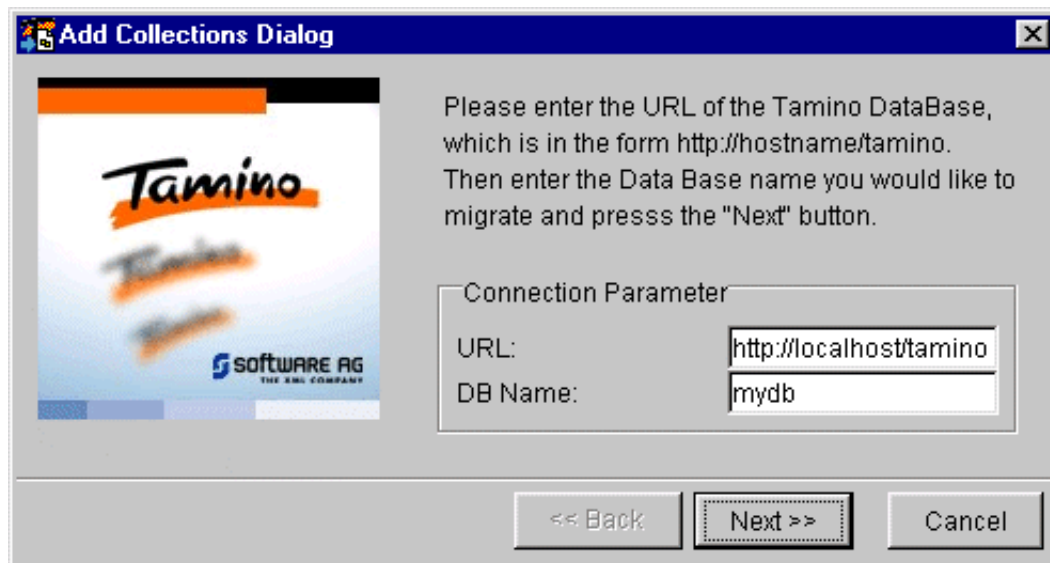
Before you start with your migration session, you should be sure that the following requirements are fulfilled:

- The database to be migrated is up (a green traffic light is visible in Tamino Manager beside the name of the database).
- Its version has been set to 3.1. (You can also check this with the Tamino Manager.)
- You have a backup of your database which you can use if required to recover to the state of the database.

The migration process of *TSD 2* schemas to *TSD 3* schemas and from data in Tamino 2.2 representation to data in Tamino 3.1 representation is started by selecting the '**Create Migration File(s)**' option from the '"File"' menu of the Tamino migration tool:



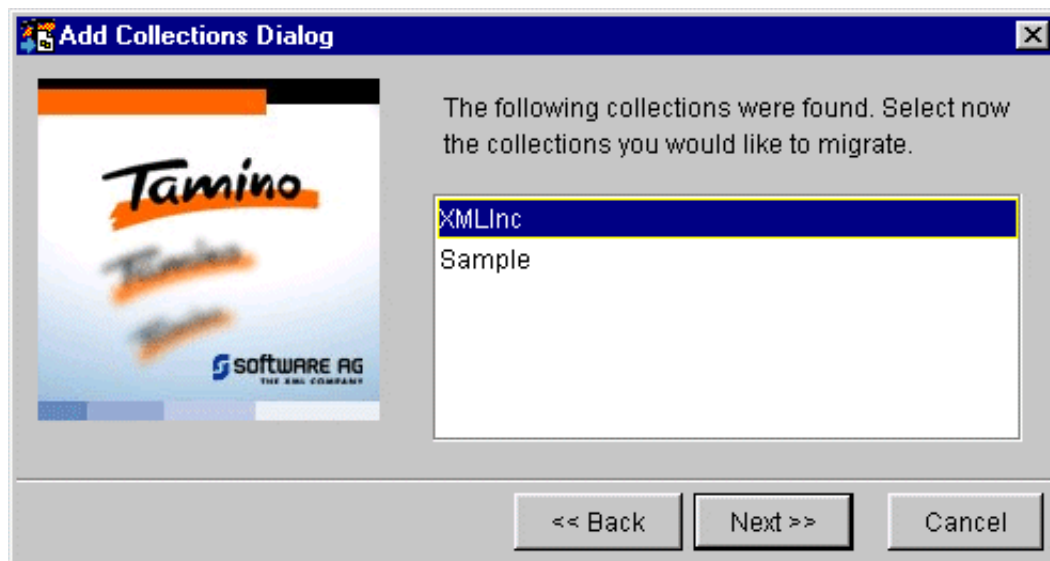
The '"Add Collections Dialog"' opens and expects the *URL* of the Tamino XML Server to connect with as input, and also the name of the database that is to be migrated.



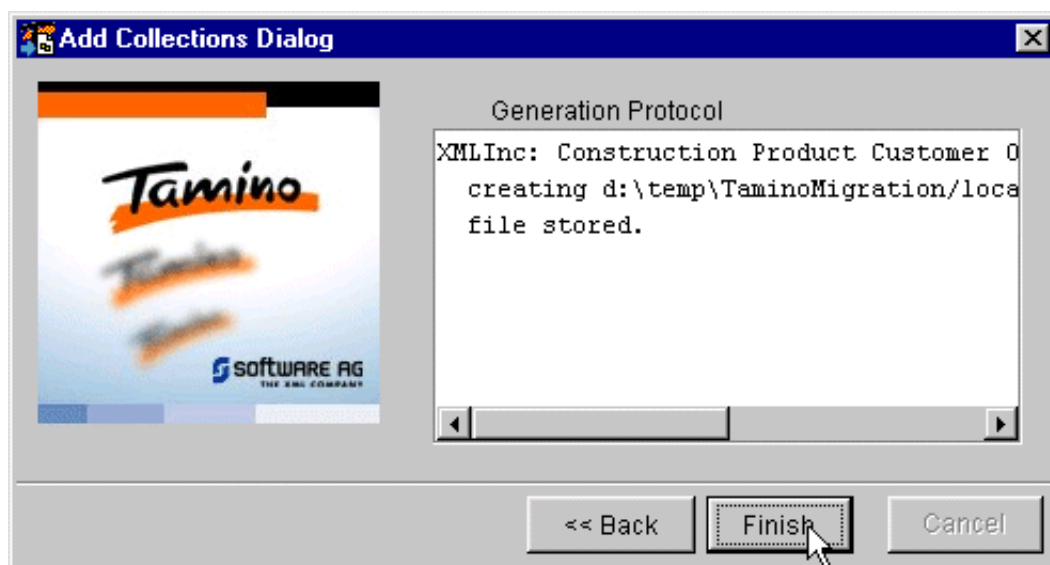
Pressing the 'Next>>' button opens a list box containing the names of all collections of the database to be migrated. You can select the collection(s) you would like to migrate.

Note:

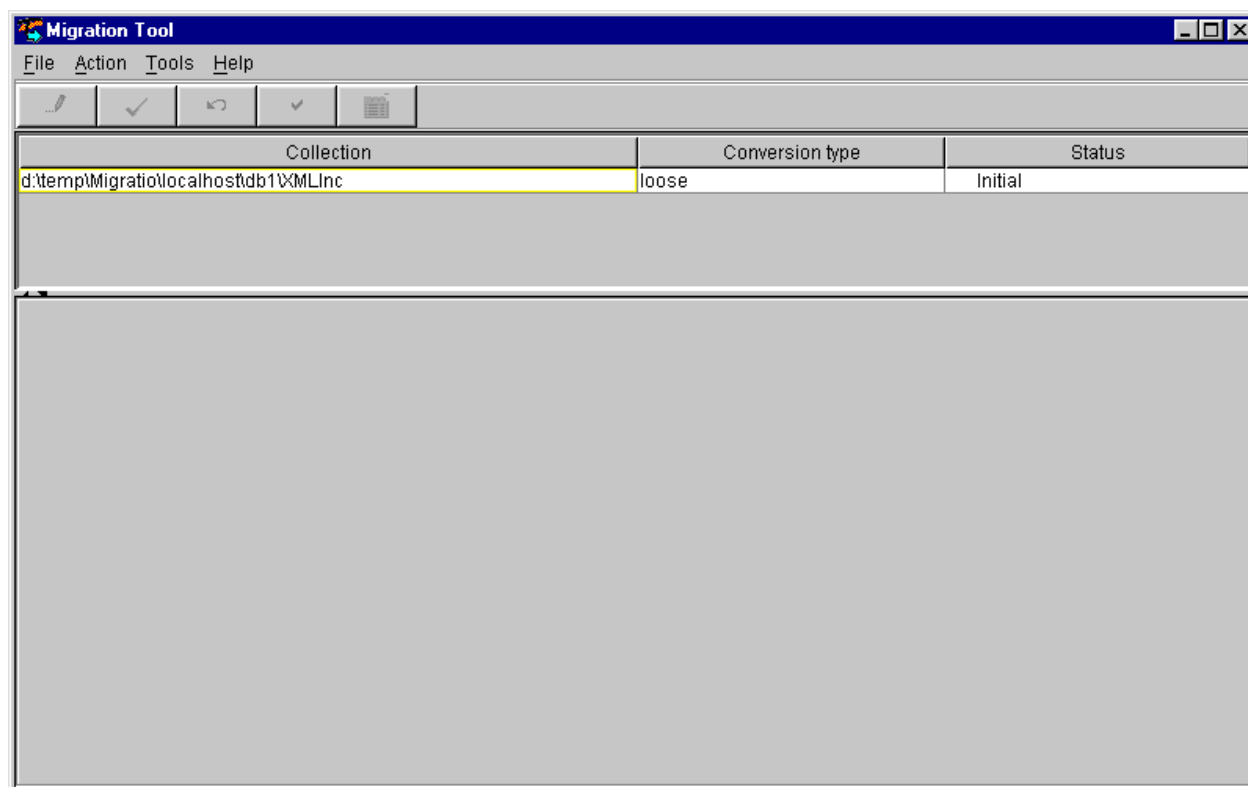
Migration is done collection-wise, i.e. only one collection can be migrated at a time. However, you may create a migration file for any number of collections from the database.



When the collection has been chosen and the 'Next>>' button has been pressed, a window showing the text of the 'Generation Protocol' appears. The 'Generation Protocol' contains information about the creation of the plan for the migration steps that are necessary for the migration of the chosen collection. This information is determined according to the results of the analysis of this collection that the migration tool now has performed. You can examine details of this later using the Task Editor of the Tamino migration tool.

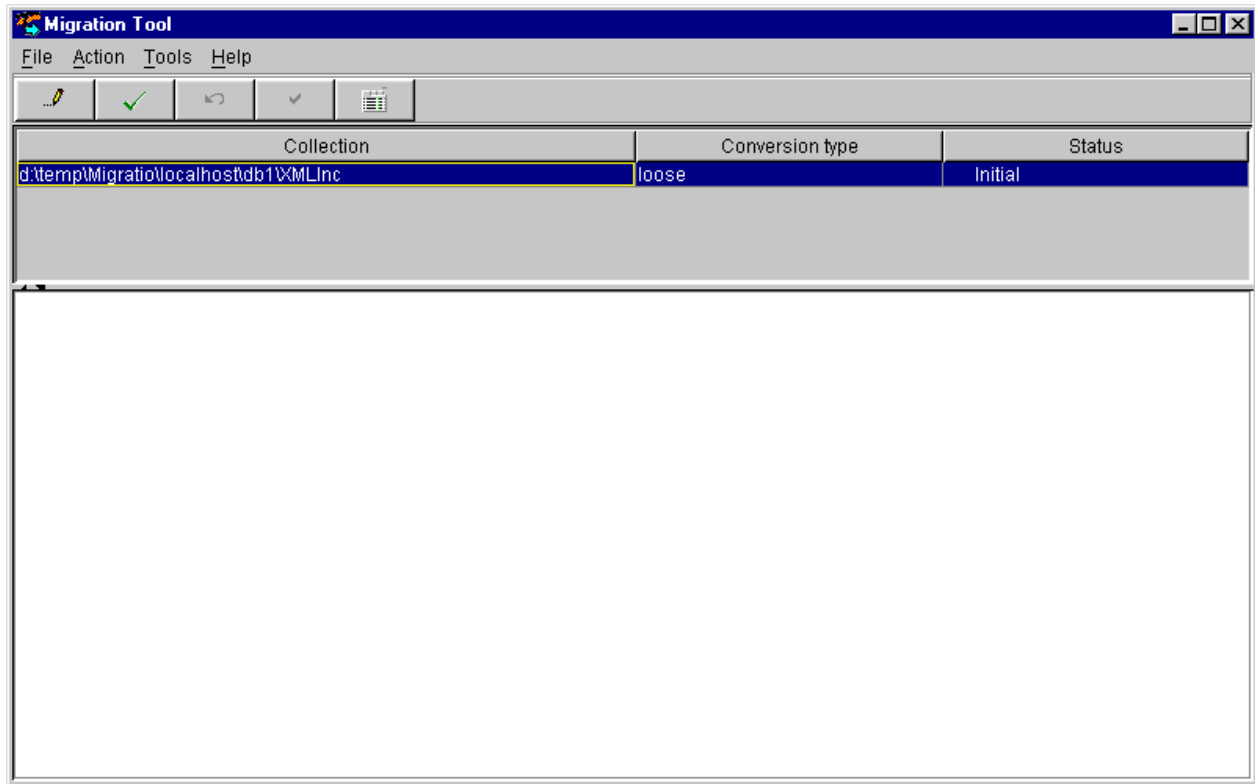


Proceed by pressing the '**Finish**' button. The main window of the Tamino migration tool will appear:



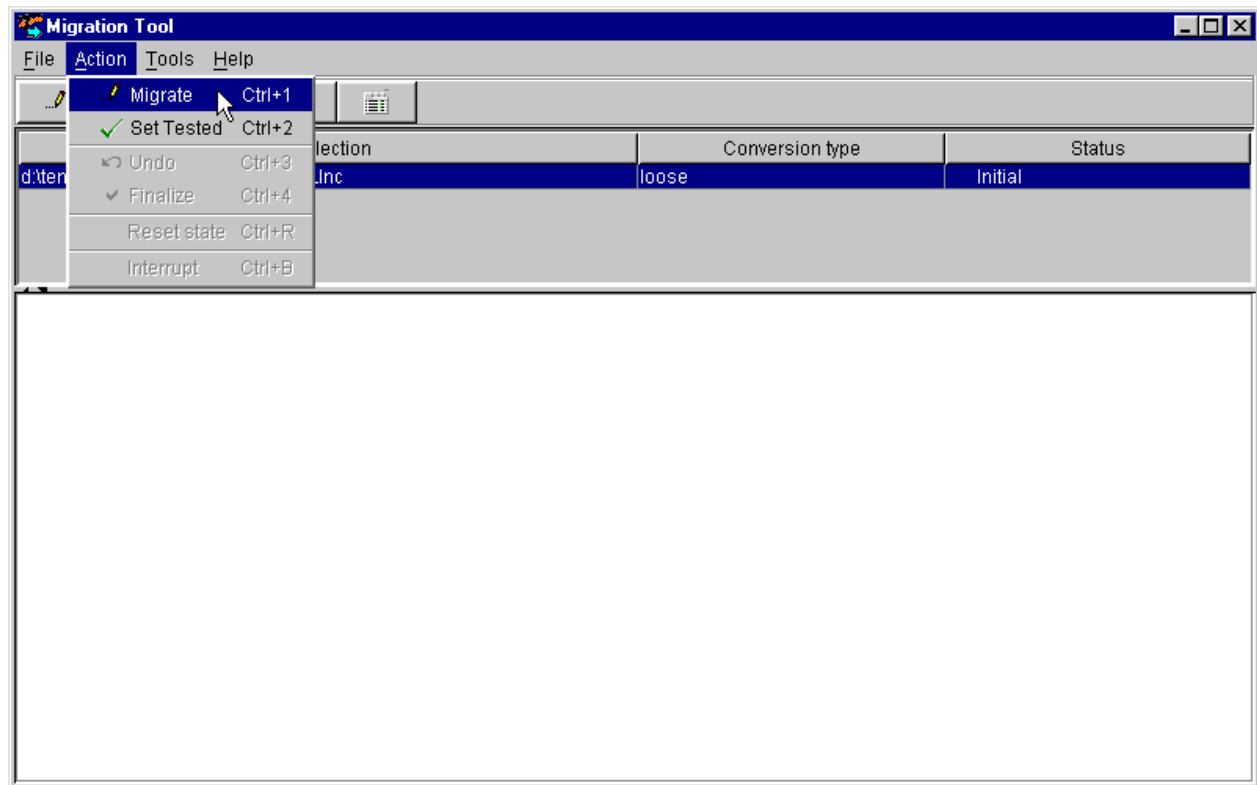
For more information about the available menu commands, please refer to the **Command Reference** of the Tamino migration tool. The upper part of the screen (below the menu and the buttons) contains a table with one line for each collection configured for migration by the migration tool. It contains entries for all collections for which the migration tool presently holds administrative information t. Because you specified information on one collection (in this example *XMLInc*), by now there is only one entry showing the name of the collection to be migrated, the current setting of the conversion type to apply during the migration of this collection (default setting is 'loose'), and the current status of the respective collection. If no activities for the migration of this collection have yet taken place, the status is set to '*Initial*'.

To proceed you should double-click on the collection to be migrated. It will then be highlighted and the screen will look like the following:

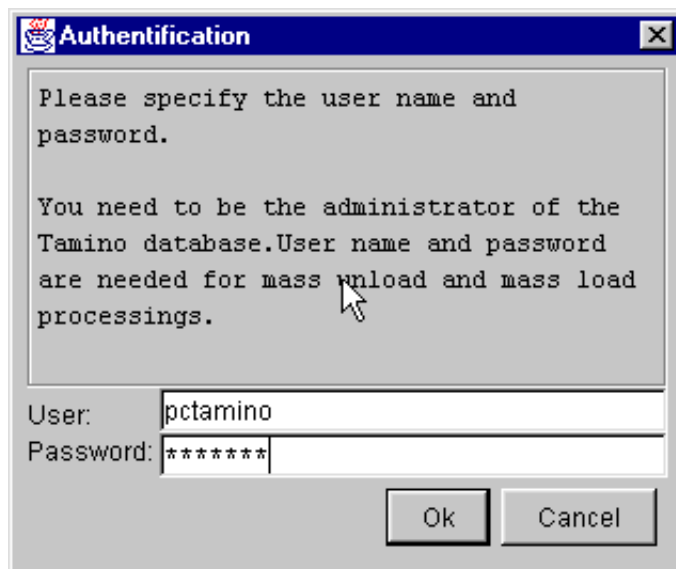


You have now finished the creation of the migration file. All information is provided and the file is generated. This also activates all menu options that apply to a collection.

You can start the migration process itself by selecting the '**Migrate**' option from the '"Action"' menu of the Tamino migration tool.



You are now prompted to provide the system with your username and password. The following box appears for this purpose:



This actually starts the migration process that performs preparations, migrates schemas and datas etc. During this process you will be informed about its progress by the protocol and status information that appears in the lower part of the window of the Tamino migration tool. If an error occurs, it will be indicated there. Another possibility to get information about the migration process is to open the Task Editor and to check whether all tasks are successfully completed.

Finally, it is necessary to decide whether or not the migration was successful after it has been performed.

► To complete the migration process of one specific collection

- In case of successful execution store the results using the **finalize** menu entry in the '"Action"' menu of the Tamino migration tool.

Caution:

This is only possible if the state of the collection has been previously set to '*Tested*'. However, setting the state of a collection to '*Tested*' is an irreversible action. Be sure that the collection has correctly migrated before setting its status to '*Tested*' in the migration tool. This is accomplished in the migration tool by the '**Set tested**' menu entry of the '"Action"' menu (or with the equivalent CTRL-2 shortcut).

Or:

In case of failure undo the results of the migration process in case of any errors in the migration protocol using the '**undo**' menu entry of the '"Action"' menu of the Tamino migration tool.

You may then try to start the migration again with a loosened schema, for instance. You can use the reset state command to reset the state of the collection to '*Initial*' and to unlock the collection, if necessary.

General Concepts of the Tamino 3.1 Migration Tool

The Tamino Migration Tool is available in 2 versions:

- as an application with a command-line interface
- with a graphical user interface (GUI version)

The Tamino Migration Tool is written in *Java*. It is contained in the package **com.softwareag.tamino.migration** and can be invoked by the following command actions:

► Invoke the Tamino Migration Tool from the command line

-

On Windows operating systems:

Goto the directory **X:\Program files\software ag\tamino\tamino <version>\migration\2xto31** (where X: is the drive where Tamino is installed and <version> the exact version number of the currently installed version). Edit the **SetEnv.cmd** script to set up the directory from the Java Runtime binary directory and the path to point to the Tamino binary location. Start the **SetEnv.cmd** to set the environment setting. **runGenerator.cmd** will generate the migration file, **runMigrator.cmd** will start the migration process after a migration file has already been produced (otherwise an error message will occur).

On UNIX operating systems:

All scripts for starting migration-related programs are located in the **bin** subdirectory of the Tamino installation directory.

These scripts are :

- **setEnv**
- **runGenerator**
- **runMigrator**

Edit the **SetEnv.cmd** script to set up the directory from the *Java* Runtime binary directory and the path to point to the Tamino binary location. Start the **SetEnv** to set the environment setting. **runGenerator** will generate the migration file, **runMigrator** will start the migration process after a migration file has already been produced (otherwise an error message will occur).

Invoke the GUI version of the Tamino Migration Tool



On Windows operating systems:

Click on the shortcut in the Windows Start Menu to invoke the GUI version of Migration Tool. Alternatively you can run the script **inomigration.cmd** from the above mentioned directory to invoke the GUI-based migration tool from the command line.

On UNIX operating systems:

The script **inomigration** that is necessary for starting the GUI version of the migration tool is also located in the **bin** subdirectory of the Tamino installation directory.

Both versions of the migration tool get their information from 2 sources about the situation on the Tamino installation to be migrated, namely

- the *Tamino Manager*
- the *XML* configuration file

Note:

The migration requires a large amount of free disk space on a single drive. It must be large enough to store the configuration data, the logging information and to provide enough space for the migration of the largest doctype in the collection to be migrated. In the GUI version the location can be set in the GUI, in the command line version the path for intermediate migration data can be set via an environment variable. For the following, assume that this path for the working folder for intermediate storage of Tamino migration data would have been adjusted to the value **c:\tmp\TaminoMigration**. This path is mentioned as the *temporary Tamino migration directory* in the following.

Caution:

The string containing the whole path to the Tamino migration files (directory name + file name) may not exceed a length of 255 characters, otherwise serious problems will occur. You should therefore prefer a short path name for the migration path. For instance, using the above mentioned default is a good idea in this context.

The Migration Process initiated by the Tamino 3.1 Migration Tool

The migration process controlled by the Tamino 3.1 Migration Tool consists of 3 phases:

- the *migration* phase
- the *test* phase
- - either the *undo* phase
 - or the *finalize* phase

The *migration* phase creates a new collection containing the migrated *TSD 3* schemas and data. After that, in a manual step (the *test* phase) the result of the *migration* phase has to be tested thoroughly (including the test of all applications working on this collection) to decide whether the result of the *migration* phase can be accepted. So this process is a semi-automatic, tool-supported procedure. Either the *undo* phase (in the case of a negative decision) or the *finalize* phase (in the case of a positive decision) is the last stage of this process. The *undo* phase will totally remove all data written to Tamino in the *migration* phase, that is the newly created collection, and restore the previous state, if necessary. The *finalize* phase will remove the original collection that was stored internally according to the rules of Tamino version 2.2. However, finalization can only take place, if all applications have been tested to work correctly. Probably many collections will be in the *test* phase simultaneously. This means that it is necessary in this situation to have precise status information. Exactly this can be delivered by the Tamino 3.1 Migration Tool.

The Tamino 3.1 Schema and Data Migration Tool - Reference for the Command Line Version

This section describes the command line scripts for the migration process under the following headings:

- The `setEnv` Script
- The `runGenerator` Script
- The `runMigrator` Script

The `setEnv` Script

The `setEnv` script makes preparations for the other scripts

- the `runGenerator` script
- the `runMigrator` script

to execute correctly.

 **To prepare for the execution of the migration scripts, you have to do the following:**

1. At first the `setEnv.cmd` file needs to be opened in an editor of your choice.

2. Edit the JRE path to show correctly to your Java Virtual Machine or Java Runtime System installation.
3. Edit the file to contain the correct path to your Tamino installation.
4. Save your edited **setEnv.cmd** file within your editor.
5. Now execute **setEnv.cmd**, or on UNIX systems the respective **setEnv** file once.

Now you are ready to execute the runGenerator script as described in the next section.

The runGenerator Script

Note:

The **runGenerator.cmd** tool is running on all Microsoft Windows operating system platforms supported by Tamino. The **runGenerator** tool is running on all *UNIX* platforms supported by Tamino. The following applies both to the **runGenerator.cmd** and to the **runGenerator** tool even if only one of both is mentioned except this is explicitly indicated otherwise.

The **runGenerator.cmd** tool and the **runGenerator** tool both have the same command line options, which are discussed within this section.

This table contains the mandatory parameters that must be specified together with **runGenerator.cmd**, or **runGenerator** respectively:

Mandatory Parameters of the runGenerator Script

Option	Meaning
-database <database name>	This parameter is necessary to specify the name of the database to be migrated from Tamino 2.2/2.3 to Tamino 3.1.
-collection <collection name>	This parameter is necessary to specify the name of the collection to be migrated from Tamino 2.2/2.3 to Tamino 3.1.

The next table contains the optional parameters that can be specified additionally to the above mentioned parameters together with **runGenerator.cmd**, or **runGenerator** respectively:

Optional Parameters of the runGenerator Script

Option	Meaning
<code>-user <user name></code>	This parameter contains the name of the user. If not specified, the default working user is assumed as default value.
<code>-password <user password></code>	This parameter contains the password for the user. If not specified, an empty string is assumed as the default value.
<code>-url <tamino url></code>	This parameter contains the URL of the Tamino XML Server to be analyzed for the generation of the migration file, for example <code>http://localhost/tamino</code> . This is needed for establishing a connection with the Tamino XML Server. The default value is <code>http://localhost/tamino</code> .
<code>-loose</code>	<p>This option enforces a loose conversion mode.</p> <p>Note: This is also the default value that is applied if neither <code>-loose</code> nor <code>-strict</code> conversion modes have been specified</p>
<code>-strict</code>	This option enforces a strict conversion mode.


The result of a run of this script is a migration file that can be processed by the **runMigrator** script that is subsequently described.

The runMigrator Script

Note:

The **runMigrator.cmd** tool is running on all Microsoft Windows operating system platforms supported by Tamino. The **runMigrator** tool is running on all *UNIX* platforms supported by Tamino. The following applies both to the **runMigrator.cmd** and to the **runMigrator** tool even if only one of both is mentioned except this is explicitly indicated otherwise.

The **runMigrator.cmd** tool and the **runMigrator** tool both have the same command line options, which are subsequently discussed.

Option	Meaning
<Migration folder>	The folder or directory where all migration files are stored. This directory can be set by runGenerator.cmd for the Windows environments, or runGenerator respectively for the <i>UNIX</i> environments.
migrate	Start the <i>migration</i> process.
setTested	Set the status to <i>tested</i> . 
undo	Start the <i>undo</i> process.
finalize	Start the <i>finalize</i> process.
reset	Reset the state of the migration file.
unlock	Unlock the collection.

Here an example for the invocation of the migration step of **runMigrator.cmd** on the Windows platform is given: You want to migrate the collection *collection1* of your database *db1*. The temporary directory was adjusted by **runGenerator.cmd** to **c:\temp\migration**. The correct invocation would then be:

```
runMigrator c:\temp\migration\localhost\db1\collection1 migrate
```

Note:

If you already specified your user name and password information when asked by **runGenerator.cmd**, you will not need to do this again during the execution of **runMigrator.cmd**. Otherwise during the execution of the **runMigrator.cmd** tool a GUI popup window requiring that information will prompt you.

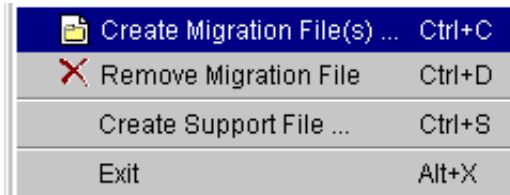
On the UNIX platforms replace **runMigrator.cmd** by **runMigrator** and use a valid *UNIX* path build according Tamino's and *UNIX*'s rules similarly to the above Windows example.

The Tamino 3.1 Schema and Data Migration Tool - Command Reference for the GUI Version

- File Menu
- Action Menu
- Tools Menu
- Help Menu
- The Task Editor

File Menu

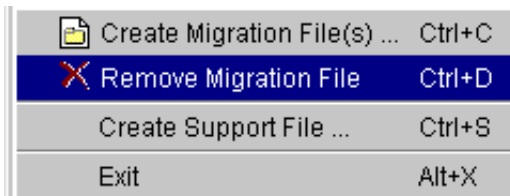
Create Migration File(s)



Shortcut: CTRL-C

This menu entry offers the possibility to create a new migration file for a single selectable collection within a selectable database on a Tamino XML Server. It opens the "Add Collection" Dialog, where you can specify both the *URL* of the Tamino XML Server you want to address and the database on this sever that contains the collection to be migrated. Having done that you will be prompted with a selection box that allows to select one collection in the selected database. When the collection is selected, a migration file containing all migration steps necessary for the complete schema and data migration of all doctypes in this collection is generated. It is then displayed in the table in the upper part of the migration tool window with its collection name, the chosen conversion type (*loose* or *strict*) and its current status which is set to *Initial* directly after generation. The contents of the migration file can then be viewed and changed with the **Task Editor**. It governs the migration process that is performed later on.

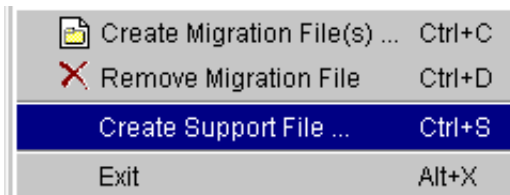
Remove Migration File



Shortcut: CTRL-D

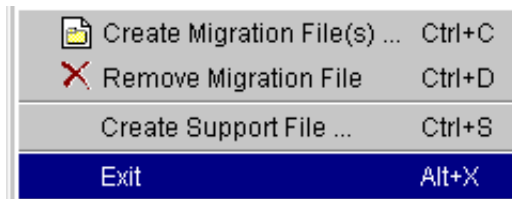
If for any reasons you want to erase a migration file, you can remove it from the migration directory using this option. However, this means that you have to perform all migration steps once again from the start.

Create Support File

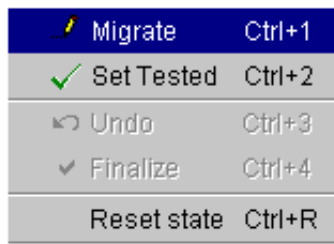


Shortcut: CTRL-S

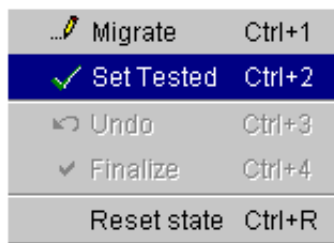
With this menu entry you can create a file containing valuable information about the current migration situation for Software AG Support in case of severe migration problems.

Exit**Shortcut: ALT-X**

This menu entry finishes the execution of the Tamino migration tool.

Action Menu**Migrate****Shortcut: CTRL-1**

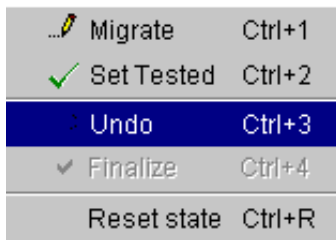
This menu entry leads you to the execution of the migration process that is comprised of the single actions that you can display and configure using the Task Editor. First it requires the input of user name and password of an account with administrative rights for authentication purposes. The series of steps is worked through until either it is successfully completed or an error occurs. In the lower area of the window of the Tamino migration tool a protocol of the actions appears that can be evaluated in case of any errors to determine the cause of execution abortions. The status indicator in the upper part of the window is also set accordingly to the result of the migration process.

Set Tested**Shortcut: CTRL-2**

After the migration of a doctype has successfully been finished you can mark this doctype as tested. Accordingly, the status in the upper part of the program window is set to '*Tested*'.

Caution:

This action cannot be undone.

Undo**Shortcut: CTRL-3**

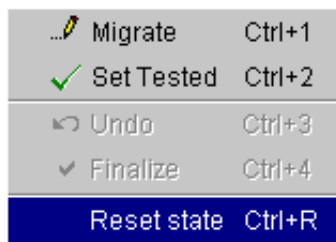
With this menu entry you can trigger the execution of the 'undo' part of the migration process that should restore the database to its original condition.

You can only use the Undo entry if either the state of the collection is indicated as '*successful*' or it has been set

Finalize**Shortcut: CTRL-4**

With this menu entry you can trigger the execution of the 'finalize' part of the migration process that definitely makes the results of the migration process permanent.

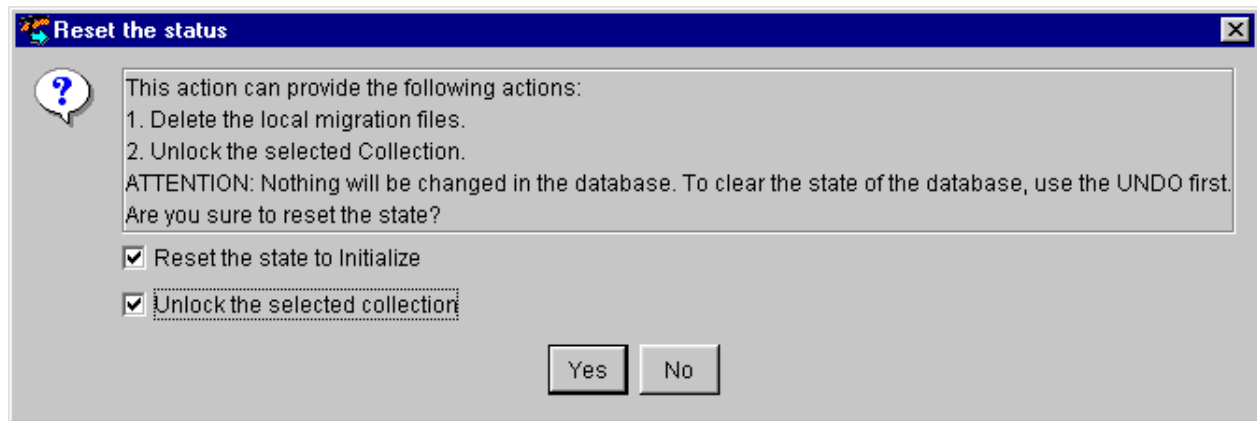
To finalize a collection it is necessary to issue a Set Tested command before the 'finalize' command can be executed. As long as this is not done, the finalize menu entry is displayed in grey instead of black.

Reset State**Shortcut: CTRL-R**

This menu entry gives you the possibility to reset the status of the migration process to '*Initial*' to be able to start the entire process again in case of errors. However, it may be necessary to additionally perform some undo operations manually depending on what migration steps actually were performed.

Reset state does not change anything in the database. Use *UNDO* first to clear the state of the database. Unlocking is also possible in conjunction with reset state.

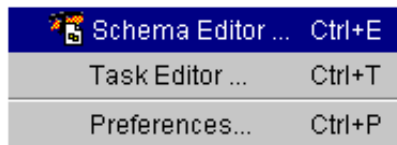
This window appears when reset state is selected:



You can determine with the check boxes whether you want to reset the state to *'Initial'* or not and whether the selected collection is unlocked or not.

Tools Menu

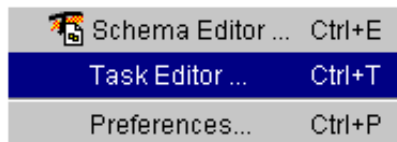
Schema Editor



Shortcut: CTRL-E

This menu entry invokes the Tamino Schema Editor, version 3, for example for manual changes in the schema definition files.

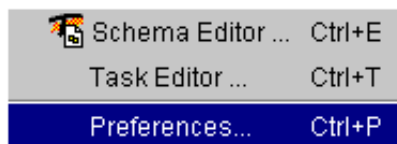
Task Editor



Shortcut: CTRL-T

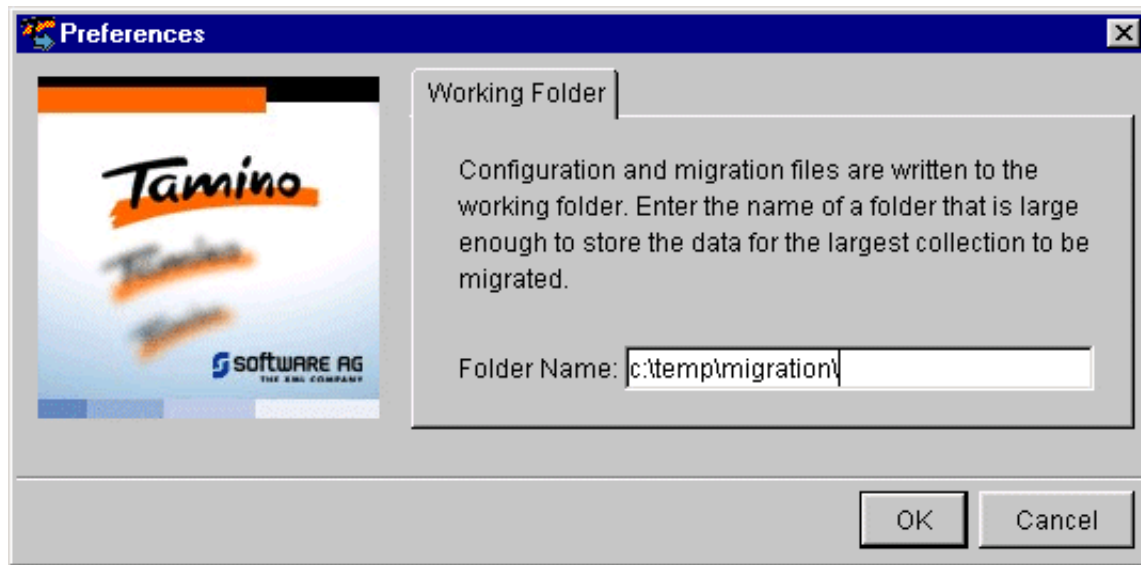
This menu entry invokes the **Task Editor**.

Preferences



Shortcut: CTRL-P

This menu entry opens the "Preferences" dialog with the sole option to change the working directory of the Tamino migration tool where it stores all its configuration and migration files. The dialog window looks like this:



This directory should be large enough to store the largest collection that you intend to migrate.



Warning:

Be sure that the whole path name (directory name + file name) of your migration files does not exceed the limit of 255 characters. It is therefore strongly recommended to use default migration path or a similarly short other migration path name. If the 255 characters limit is violated severe problems will occur. This applies both to Windows and to UNIX platforms.

Help Menu

About



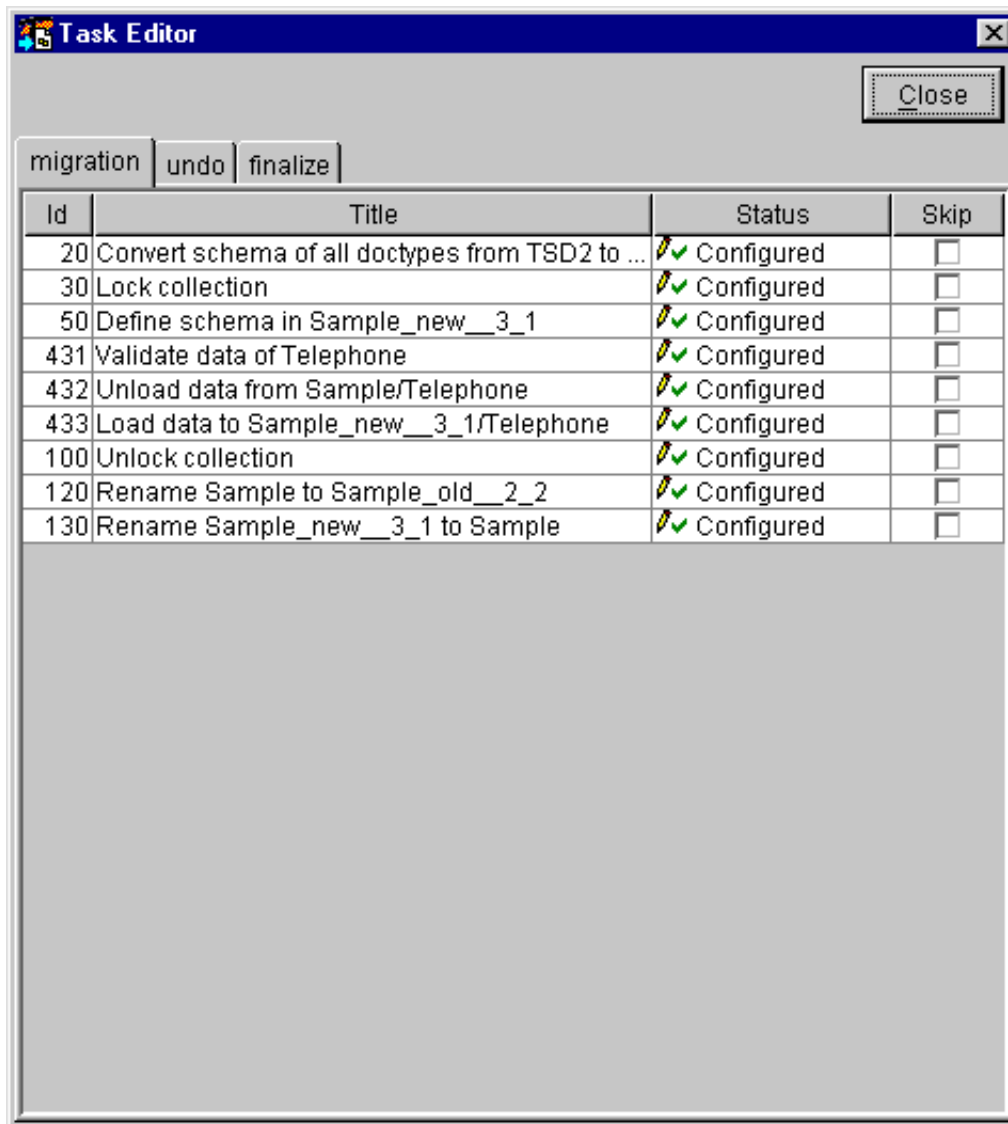
Shortcut: CTRL-A

This menu entry opens a box displaying information about the Tamino migration tool.

The Task Editor

The **Task Editor** allows you to display and to configure the tasks for migration, undo and finalize phase of the migration process without having to edit the configuration file manually. Especially it is possible to skip a step in the process that does not work to proceed with the others.

The Task Editor is invoked with the Task Editor (or the CTRL-E shortcut on the keyboard). It looks like this:



The window shows a list providing the following information about each single step in the *migration*, *undo* and *finalize* stage of the migration process:

1. The ID of the migration step.

Note:

These ID values are needed by the Task Editor and the migration tool to identify the single tasks uniquely. They are automatically generated by the migration tool during processing.



Warning:

The user must not modify these IDs. Otherwise severe problems in the migration process will occur most probably.

2. The title of the migration step containing a very brief description of what the step actually does.
3. The current status of the step (configured, executed, ...)
4. A checkbox that can be marked to indicate the migration tool that the corresponding migration step has to be skipped during the execution of the migration process.

You can switch between the display of the list of the *migration*, *undo* and *finalize* stage of the migration process by selecting the respective tab in the upper left corner of the Task Editor window.

As long as the Task Editor window is open the migration tool window is inactive. To return to the Task Editor window, you must close the window by clicking the **Close** button or pressing ALT-C on the keyboard.

Technical Details of the Tamino Migration Tool

This document describes the following technical aspects of the Tamino Migration Tool:

- General Information
- The Required Input for the Tamino 3.1 Migration Tool
- The Generated Output of the Tamino 3.1 Migration Tool
- The Configuration File for the Tamino 3.1 Migration Tool
- Special Hints for Troubleshooting

The migration process is generally initiated in the following way:

To configure and run the migration process

You must have a correctly running Java Virtual Machine (JVM) or Java Run-time Environment (JRE) available. The path and classpath environment variables must be set accordingly to point to this JVM/JRE.

1. Run the Session Generator using the script file **runGenerator.cmd** by calling the command `runGenerator.cmd`.

This will start the Session Generator and produce an *XML* configuration file for the use of the task manager in your configured output directory, for example in **C:\tmp\TaminoMigration**. If you run this in a command line session at the prompt, you will read the filename at the **stdout** console. This filename is referred as `<filename>` in the rest of these instructions.

Caution:

The string containing the whole path to the Tamino migration files (directory name + file name) may not exceed a length of 255 characters, otherwise serious problems will occur. You should therefore prefer a short path name for the migration path. For instance, using the above mentioned default is a good idea in this context.

2. Using this file name of the newly generated *XML* configuration file run the "Tamino" 3.1 Migration Tool via the script **runMigrator.cmd** using the command line `runMigrator <filename>`.

This starts the processing of the tasks by the Migrator and the Task Manager.

The next subsections of this document explain what temporary files the migration tool produces and how to setup configuration files for the various possible scenarios that may occur in Tamino migration to Tamino version 3.1.

General Information

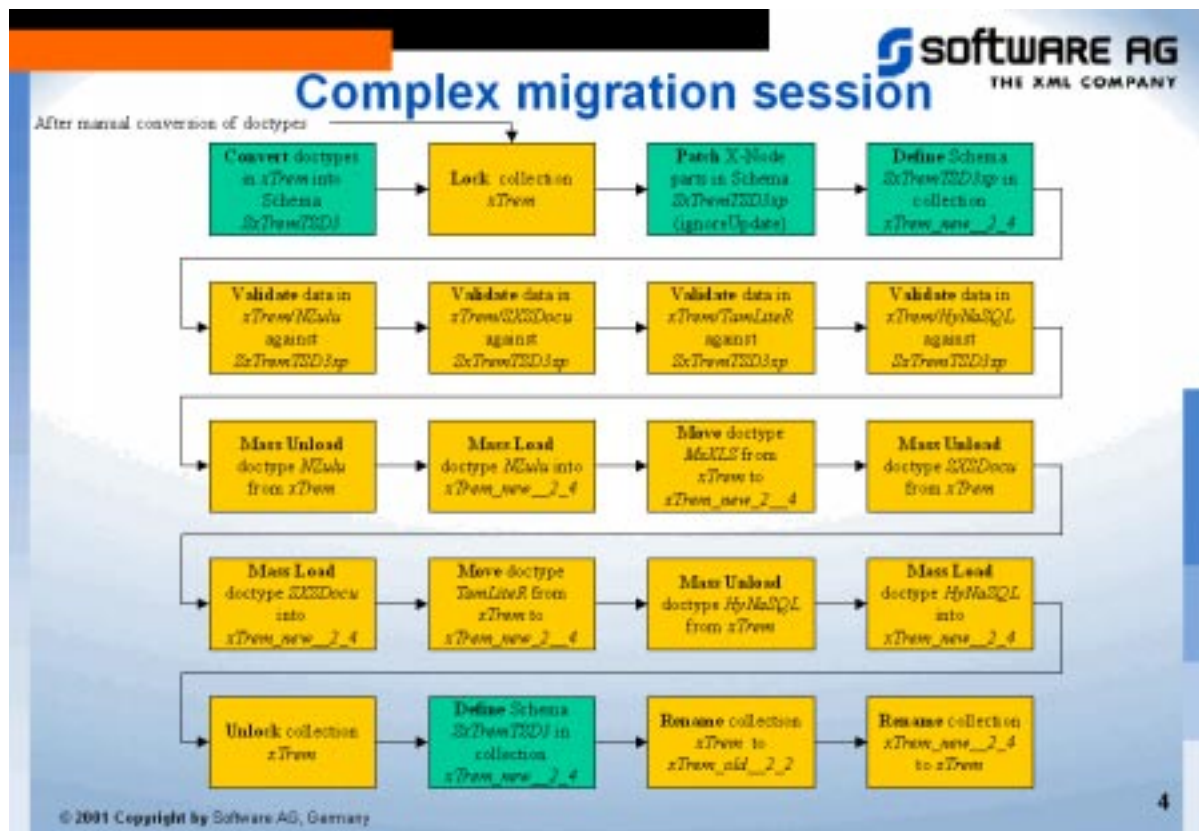
The following 3 pictures describe a complex migration scenario in a quite realistic situation. This is divided into the following 3 sessions

- the *migration* session (which corresponds to the '*Migrate*' phase)

All steps of the migration process that are presented here as a single box within this diagram correspond to a task (i.e. a single line in the **Task Editor** of the Tamino migration tool). Typical operations that the Tamino migration tool can be configured for are:

- Conversion of doctype definition from TSD 2 to TSD 3
- Locking of collections
- Unlocking of collections
- Adapt the schema for information stored outside of Tamino with X-Node.
- Define Schema operations
- Validations against XML Schema
- Mass loading operations using the Tamino Data Loader
- Mass unloading operations using the Tamino Data Loader
- Moving doctypes
- Renaming collections

An example showing all those possibilities is shown in the figure below:



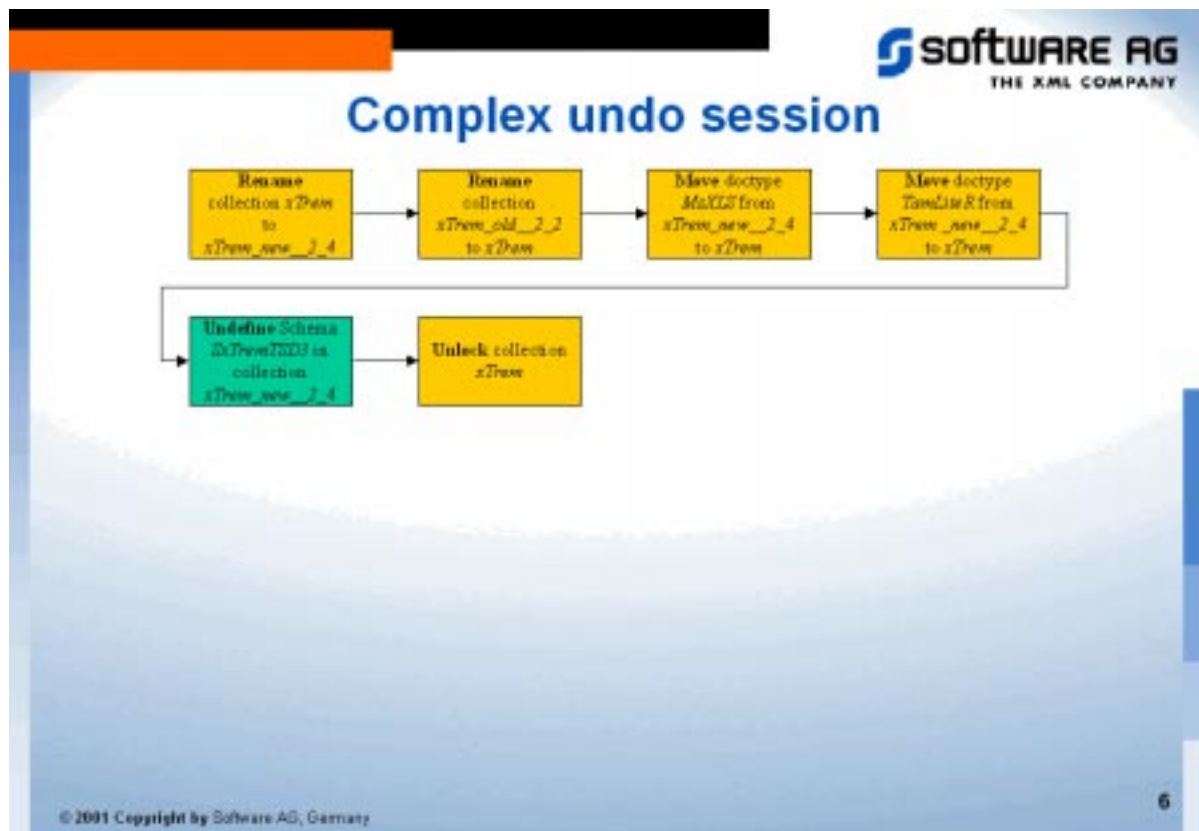
- the *undo* session (which corresponds to the 'Undo' phase)

The Tamino migration tool automatically creates a sequence of Tasks that are suitable for undoing all the changes that have been made during the migration process and completely restoring the state prior to the invocation of the migration tool. In the Task Editor this is shown in the 'undo' pane.

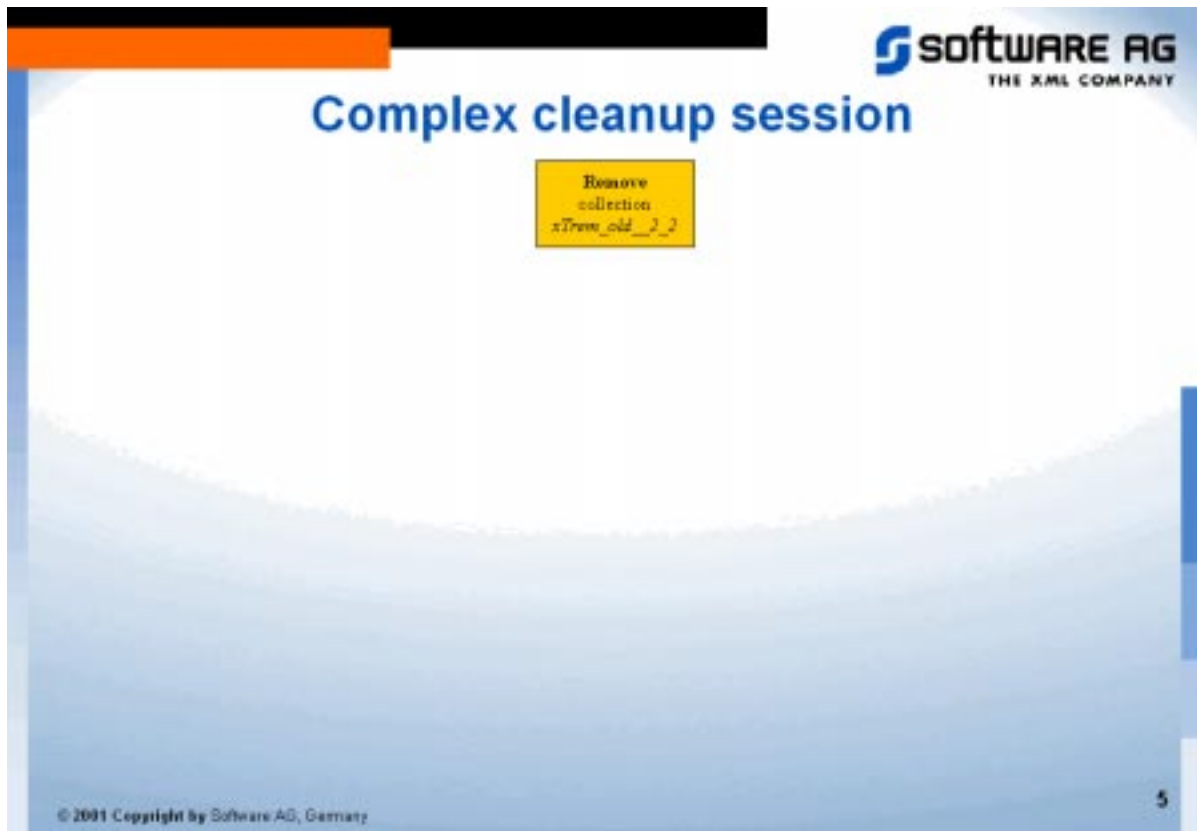
Applicable steps in the *undo* session include:

- Moving doctypes
- Renaming collections
- Unlocking of collections
- undefining schemas

An example showing all of those possibilities is shown in the picture.



- the *cleanup* session (which corresponds to the 'Finalize' phase). It contains all the steps of the migration process which are typically needed to ensure that the result of the migration process replaces the old data that have been migrated and that no erroneous data remain on the disk.



The Required Input for the "Tamino" 3.1 Migration Tool

For the Tamino 3.1 Migration Tool, at least the following input information is required:

- The user name.
- The user's password.
- The *URL* of the Tamino installation to be migrated.
- The database name, and, if not all collections within this database are to be migrated, the respective collection names (this information may be selected within the GUI version of the Tamino 3.1 Migration Tool)
- The intended conversion mode for schema conversion (*loose* or *strict* conversion, this is specified using the *XML* configuration file)
- Whether rejects of the Tamino Data Loader are tolerated or not in the data conversion (also specified within the *XML* configuration file)

The Generated Output of the "Tamino" 3.1 Migration Tool

The Tamino 3.1 Migration Tool produces a set of files for each collection it transfers from Tamino 2.2 to Tamino 3.1. It first generates a directory structure below the *temporary Tamino migration directory* in this manner:

In the *temporary Tamino migration directory* a subdirectory with the name of a the host computer on which the collection to be migrated resides is created. In this directory, a subdirectory with the name of the database to which the collection to be migrated belongs is created. In this directory, a subdirectory with the name of the collection to be migrated itself is created. This directory will contain the following files and directories after the migration:

1. The file **log.xml** or **log.txt** in the **config** subdirectory: These files contain logging information on the proceeding migration.
2. The **schema** subdirectory will contain the whole schema information of the migrated doctype.
3. The **data** subdirectory will contain the whole *XML* data of the migrated doctype.
4. The file **migration.xml** in the **config** subdirectory : This is the original control file for the migration of one collection.

The **config** directory also contains some other files used for the management of the migration tool.

The Configuration File for the "Tamino" 3.1 Migration Tool

The configuration file for the Tamino 3.1 Migration Tool is based on single tasks that are represented in *XML* notation. A task in this context (referred to as *XML* task in the following) consists of an *XML* code fragment like this:

```
<task>
  <title>initial task. Get and save input needed by all tasks</>
  <skip>false</>
  <ID>10</>
<className>com.softwareag.tamino.migration.control.tasks.CommonConfig</>
  <status>0</>
  <input>
    <ID>1</>
    <name>taminoUrl</>
  </input>
  <input>
    <ID>2</>
    <name>databaseName</>
  </input>
</task>
```

Accordingly, the Tamino 3.1 Migration Tool deals with tasks in this context. Its central component is a task manager that is able to administer and execute such tasks. The meaning of the single elements of the *XML* tasks are explained in the following table:

Tag	Description	Value
ID	The identification if the task	Integer
title	The title of the task	String
className	The class name responsible for configuration, execution and rollback methods. This represents the "RuntimeClass". It is given in Java form.	String
skip	If this task has to be skipped during configuration or execution, the value must be <i>true</i> .	<i>true/false</i>
nextTask	The ID of the next task in the chain	Integer
status	The current status of the task	Integer
duration	The time duration of the task	
input	Contains all input values required by the task during execution. Usually set by the configuration	Vector
output	Values produced by the execution.	Vector

Each *XML* task instance refers to a Runtime Class specified by the *className* tag of the *XML* task. Logically, the *XML* file can model more than one chain of tasks. This is handled by specifying an *Execution Path* in that file.

Here is a realistic example for a configuration file for a migration situation from Tamino 2.2 to Tamino 3.1, this means, including both schema and data migration. For reference only, the XML file is marked with line numbers on the left side which of course must be omitted in practical use.

```

0010 <?xml version="1.0" encoding="UTF-8"?>
0020 <session>
0030 <title>migration of http://localhost/tamino/myDatabase/tct</title>
0040 <host>PCUSER</host>
0050 <user>UserName</user>
0060 <workingFolder>c:\tmp\workingFolder>
0070 <executionPath>
0080 <name>migration</name>
0090 <status>1</status>
0100 <initialTask>10</initialTask>
0110 <firstTask>20</firstTask>
0120 </executionPath>
0130 <executionPath>
0140 <name>cleanup</name>
0150 <status>1</status>
0160 <initialTask>10</initialTask>
0170 <firstTask>300</firstTask>
0180 </executionPath>
0190 <executionPath>
0200 <name>undo</name>
0210 <status>1</status>
0220 <initialTask>10</initialTask>
0230 <firstTask>400</firstTask>
0240 </executionPath>
0250 <task>
0260 <ID>10</ID>
0270 <title>initial task. Get and save input needed by all tasks</title>
0280 <className>com.softwareag.tamino.migration.control.tasks.CommonConfig</className>
0290 <skip>false</skip>
0300 <status>1</status>
0310 <input>
0320 <ID>1</ID>
0330 <name>taminoUrl</name>
0340 <value>http://localhost/tamino</value>
0350 </input>
0360 <input>
0370 <ID>2</ID>
0380 <name>databaseName</name>
0390 <value>myDatabase</value>
0400 </input>
0410 <input>
0420 <ID>3</ID>
0430 <name>collectionName</name>
0440 <value>tct</value>
0450 </input>
0460 <input>
0470 <ID>4</ID>
0480 <name>conversionOption</name>
0490 <value>loose</value>
0500 </input>
0510 <input>
0520 <ID>5</ID>
0530 <name>sessionKey</name>
0540 <value>dummy</value>
0550 </input>
0560 <input>
0570 <ID>6</ID>
0580 <name>sessionID</name>
0590 <value>dummy</value>
0600 </input>
0610 <input>
0620 <ID>7</ID>

```

```
0630     <name>userName</name>
0640     <value>defaultuser</value>
0650   </input>
0660   <input>
0670     <ID>8</ID>
0680     <name>password</name>
0690     <value>.</value>
0700   </input>
0710 </task>
0720 <task>
0730   <ID>20</ID>
0740   <title>Convert schema of all doctypes from TSD2 to TSD3</title>
0750   <className>com.softwareag.tamino.migration.control.tasks.ConvertSchema</className>
0760   <skip>false</skip>
0770   <status>1</status>
0780   <nextTask>30</nextTask>
0790   <input>
0800     <ID>1</ID>
0810     <name>collectionName</name>
0820     <value>tct</value>
0830   </input>
0840   <input>
0850     <ID>2</ID>
0860     <name>schemaName</name>
0870     <value>tct</value>
0880   </input>
0890   <input>
0900     <ID>3</ID>
0910     <name>doctypeNames</name>
0920     <value>LoggingOfServices</value>
0930   </input>
0940   <input>
0950     <ID>4</ID>
0960     <name>destination</name>
0970     <value>localhost/myDatabase/tct/schema/tct.xml</value>
0980   </input>
0990   <input>
1000     <ID>6</ID>
1010     <name>deleteFiles</name>
1020     <value>false</value>
1030   </input>
1040 </task>
1050 <task>
1060   <ID>30</ID>
1070   <title>Lock collections</title>
1080   <className>com.softwareag.tamino.migration.control.tasks.LockCollection</className>
1090   <skip>false</skip>
1100   <status>1</status>
1110   <nextTask>50</nextTask>
1120   <input>
1130     <ID>1</ID>
1140     <name>collectionName</name>
1150     <value>tct</value>
1160   </input>
1170 </task>
1180 <task>
1190   <ID>50</ID>
1200   <title>Define schema in tct_new_3_1</title>
1210   <className>com.softwareag.tamino.migration.control.tasks.DefineSchema</className>
1220   <skip>false</skip>
1230   <nextTask>431</nextTask>
1240   <input>
1250     <ID>1</ID>
1260     <name>source</name>
1270     <value>localhost/myDatabase/tct/schema/tct.xml</value>
1280   </input>
1290   <input>
1300     <ID>2</ID>
1310     <name>patchCollectionName</name>
1320     <value>tct_new_3_1</value>
1330   </input>
1340 </task>
1350 <task>
1360   <ID>100</ID>
1370   <title>Unlock collection</title>
1380   <className>com.softwareag.tamino.migration.control.tasks.UnlockCollection</className>
1390   <skip>false</skip>
1400   <status>1</status>
1410   <nextTask>120</nextTask>
1420   <input>
1430     <ID>1</ID>
1440     <name>collectionName</name>
1450     <value>tct</value>
1460   </input>
1470 </task>
1480 <task>
1490   <ID>120</ID>
1500   <title>Rename tct to tct_old_2_2</title>
1510   <className>com.softwareag.tamino.migration.control.tasks.RenameCollection</className>
1520   <skip>false</skip>
1530   <status>1</status>
1540   <nextTask>130</nextTask>
1550   <input>
1560     <ID>1</ID>
1570     <name>oldCollectionName</name>
1580     <value>tct</value>
1590   </input>
1600   <input>
1610     <ID>2</ID>
1620     <name>newCollectionName</name>
1630     <value>tct_old_2_2</value>
1640   </input>
1650 </task>
1660 <task>
1670   <ID>130</ID>
1680   <title>Rename tct_new_3_1 to tct</title>
1690   <className>com.softwareag.tamino.migration.control.tasks.RenameCollection</className>
1700   <skip>false</skip>
1710   <status>1</status>
1720   <input>
1730     <ID>1</ID>
1740     <name>oldCollectionName</name>
1750     <value>tct_new_3_1</value>
1760   </input>
1770   <input>
1780     <ID>2</ID>
1790     <name>newCollectionName</name>
1800     <value>tct</value>
1810   </input>
1820 </task>
1830 <task>
1840   <ID>300</ID>
1850   <title>Remove collection tct_new_3_1</title>
1860   <className>com.softwareag.tamino.migration.control.tasks.RemoveCollection</className>
1870   <skip>false</skip>
1880   <status>1</status>
1890   <duration>
1900     <start></start>
```

```
1920 <end></end>
1930 </duration>
1940 <input>
1950 <ID>1</ID>
1960 <name>collectionName</name>
1970 <value>tct_new__3_1</value>
1980 </input>
1990 <input>
2000 <ID>2</ID>
2010 <name>deleteNonEmpty</name>
2020 <value>true</value>
2030 </input>
2040 </task>
2050 <task>
2060 <ID>400</ID>
2070 <title>Restore collection name</title>
2080 <className>com.softwareag.tamino.migration.control.tasks.RestoreCollectionNames</className>
2090 <skip>false</skip>
2100 <status>1</status>
2110 <nextTask>420</nextTask>
2120 <duration>
2130 <start></start>
2140 <end></end>
2150 </duration>
2160 <input>
2170 <ID>1</ID>
2180 <name>oldCollectionName</name>
2190 <value>tct_old__2_2</value>
2200 </input>
2210 <input>
2220 <ID>2</ID>
2230 <name>newCollectionName</name>
2240 <value>tct_new__3_1</value>
2250 </input>
2260 <input>
2270 <ID>3</ID>
2280 <name>prodCollectionName</name>
2290 <value>tct</value>
2300 </input>
2310 </task>
2320 <task>
2330 <ID>420</ID>
2340 <title>Undefine schema of tct_new__3_1</title>
2350 <className>com.softwareag.tamino.migration.control.tasks.UndefineSchema</className>
2360 <skip>false</skip>
2370 <status>1</status>
2380 <nextTask>430</nextTask>
2390 <input>
2400 <ID>1</ID>
2410 <name>collectionName</name>
2420 <value>tct_new__3_1</value>
2430 </input>
2440 <input>
2450 <ID>2</ID>
2460 <name>schemaName</name>
2470 <value>tct</value>
2480 </input>
2490 </task>
2500 <task>
2510 <ID>430</ID>
2520 <title>Remove collection tct_new__3_1</title>
2530 <className>com.softwareag.tamino.migration.control.tasks.RemoveCollection</className>
2540 <skip>false</skip>
2550 <status>1</status>
2560 <input>
2570 <ID>1</ID>
2580 <name>collectionName</name>
2590 <value>tct_new__3_1</value>
2600 </input>
2610 <input>
2620 <ID>2</ID>
2630 <name>deleteNonEmpty</name>
2640 <value>false</value>
2650 </input>
2660 </task>
2670 <task>
2680 <ID>431</ID>
2690 <title>Validate data of LoggingOfServices</title>
2700 <className>com.softwareag.tamino.migration.control.tasks.ValidateDoctype</className>
2710 <skip>false</skip>
2720 <status>1</status>
2730 <nextTask>432</nextTask>
2740 <input>
2750 <ID>1</ID>
2760 <name>doctypeName</name>
2770 <value>LoggingOfServices</value>
2780 </input>
2790 <input>
2800 <ID>2</ID>
2810 <name>dataCollectionName</name>
2820 <value>tct</value>
2830 </input>
2840 <input>
2850 <ID>3</ID>
2860 <name>checkCollectionName</name>
2870 <value>tct_new__3_1</value>
2880 </input>
2890 </task>
2900 <task>
2910 <ID>432</ID>
2920 <title>Unload data from tct/LoggingOfServices</title>
2930 <className>com.softwareag.tamino.migration.control.tasks.MassUnload</className>
2940 <skip>false</skip>
2950 <status>1</status>
2960 <nextTask>433</nextTask>
2970 <input>
2980 <ID>1</ID>
2990 <name>collectionName</name>
3000 <value>tct</value>
3010 </input>
3020 <input>
3030 <ID>2</ID>
3040 <name>doctypeName</name>
3050 <value>LoggingOfServices</value>
3060 </input>
3070 <input>
3080 <ID>3</ID>
3090 <name>destination</name>
3100 <value>localhost/myDatabase/tct/data/LoggingOfServices.xml</value>
3110 </input>
3120 </task>
3130 <task>
3140 <ID>433</ID>
3150 <title>Load data to tct_new__3_1/LoggingOfServices</title>
3160 <className>com.softwareag.tamino.migration.control.tasks.MassLoad</className>
3170 <skip>false</skip>
3180 <status>1</status>
3190 <nextTask>100</nextTask>
3200 <input>
```

```

3210      <ID>1</ID>
3220      <name>collectionName</name>
3230      <value>tct_new__3_1</value>
3240    </input>
3250    <input>
3260      <ID>2</ID>
3270      <name>docTypeName</name>
3280      <value>LoggingOfServices</value>
3290    </input>
3300    <input>
3310      <ID>3</ID>
3320      <name>source</name>
3330      <value>localhost/myDatabase/tct/data/LoggingOfServices.xml</value>
3340    </input>
3350    <input>
3360      <ID>4</ID>
3370      <name>tolerateRejects</name>
3380      <value>true</value>
3390    </input>
3400    <input>
3410      <ID>5</ID>
3420      <name>rejectsFile</name>
3430      <value>localhost/myDatabase/tct/data/LoggingOfServices_REJECTS.xml</value>
3440    </input>
3450    <input>
3460      <ID>6</ID>
3470      <name>deleteFiles</name>
3480      <value>true</value>
3490    </input>
3500  </task>
3510 </session>

```

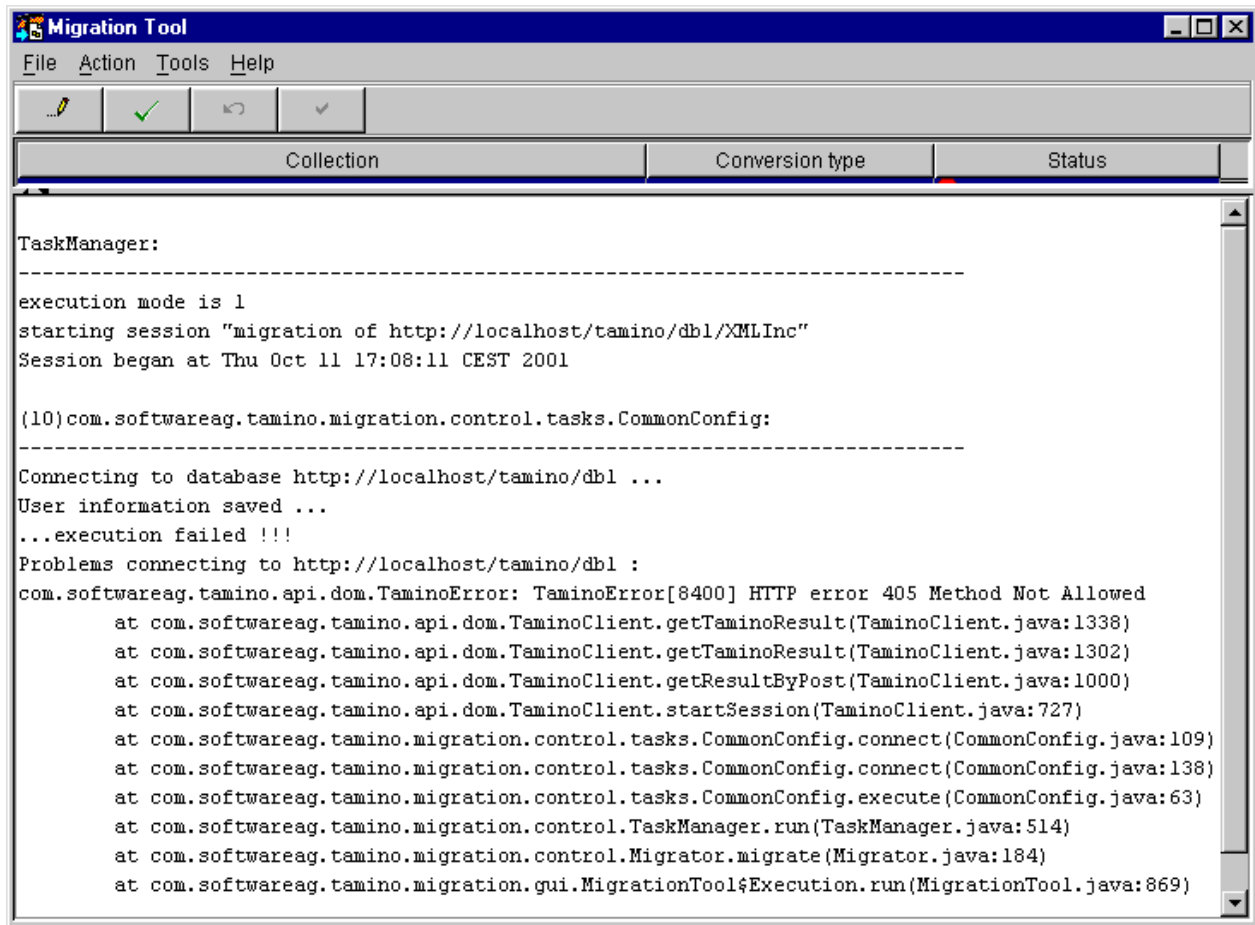
The following scenario could make it necessary to edit this file manually. It is recommended to use the Task Editor that can be invoked from the GUI interface of the Migration Tool to edit the tasks.

The Validate-step failed.

This could happen if the *TSD3* schema generated from the converter is too strict. In this case call the Tamino Schema Editor to edit the generated *TSD3* schema and define it manually. After the schema is modified the step in line 2700 can be skipped. Change the value in line 2710 to true to skip this step and repeat the migration process from the load step.

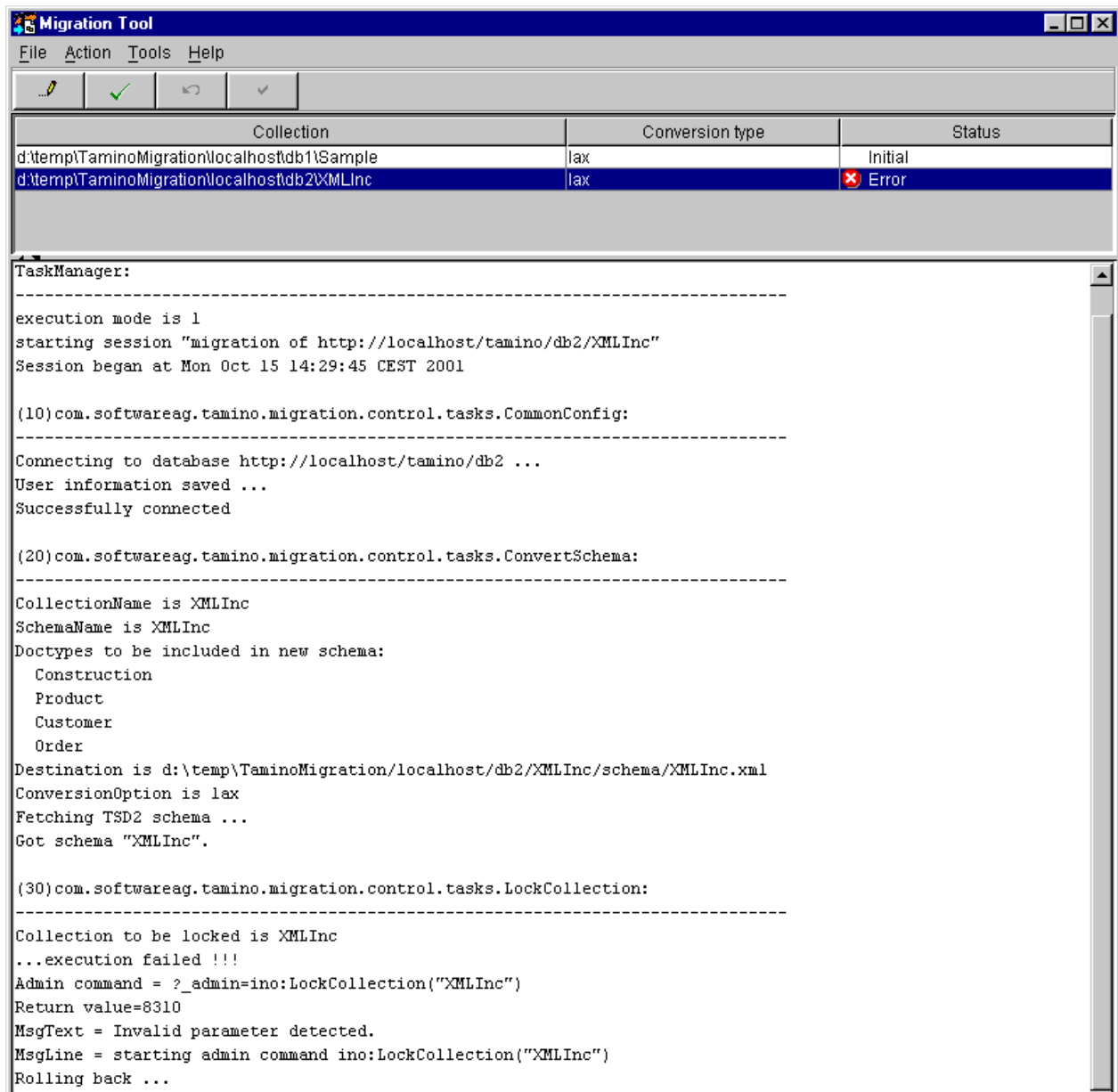
Special Hints for Troubleshooting

Tamino Error 8400



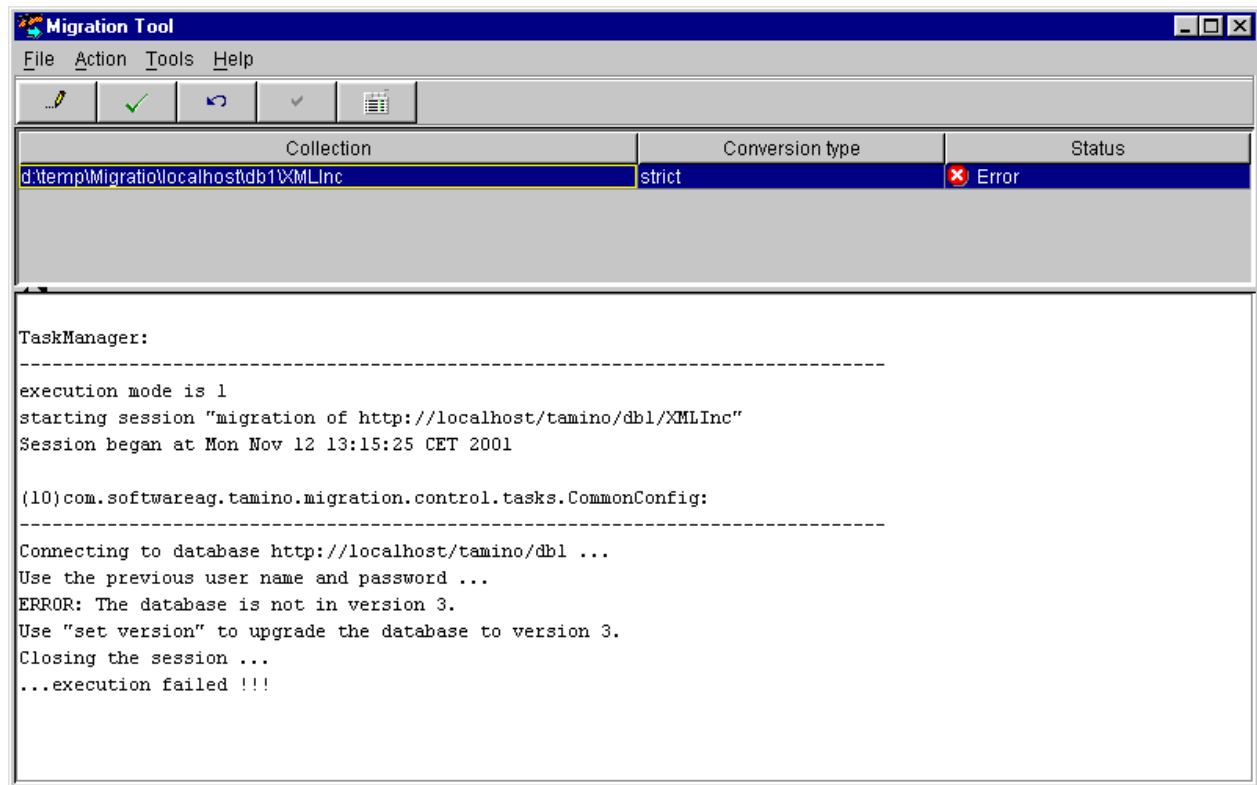
If you receive a Tamino error 8400 (HTTP error 405) in step (10) in module **com.softwareag.tamino.migration.control.tasks.CommonConfig**, this is caused by the fact that the database has not been started. Start the database using the Tamino Manager, and you may proceed.

Invalid parameter detected. Return value=8310



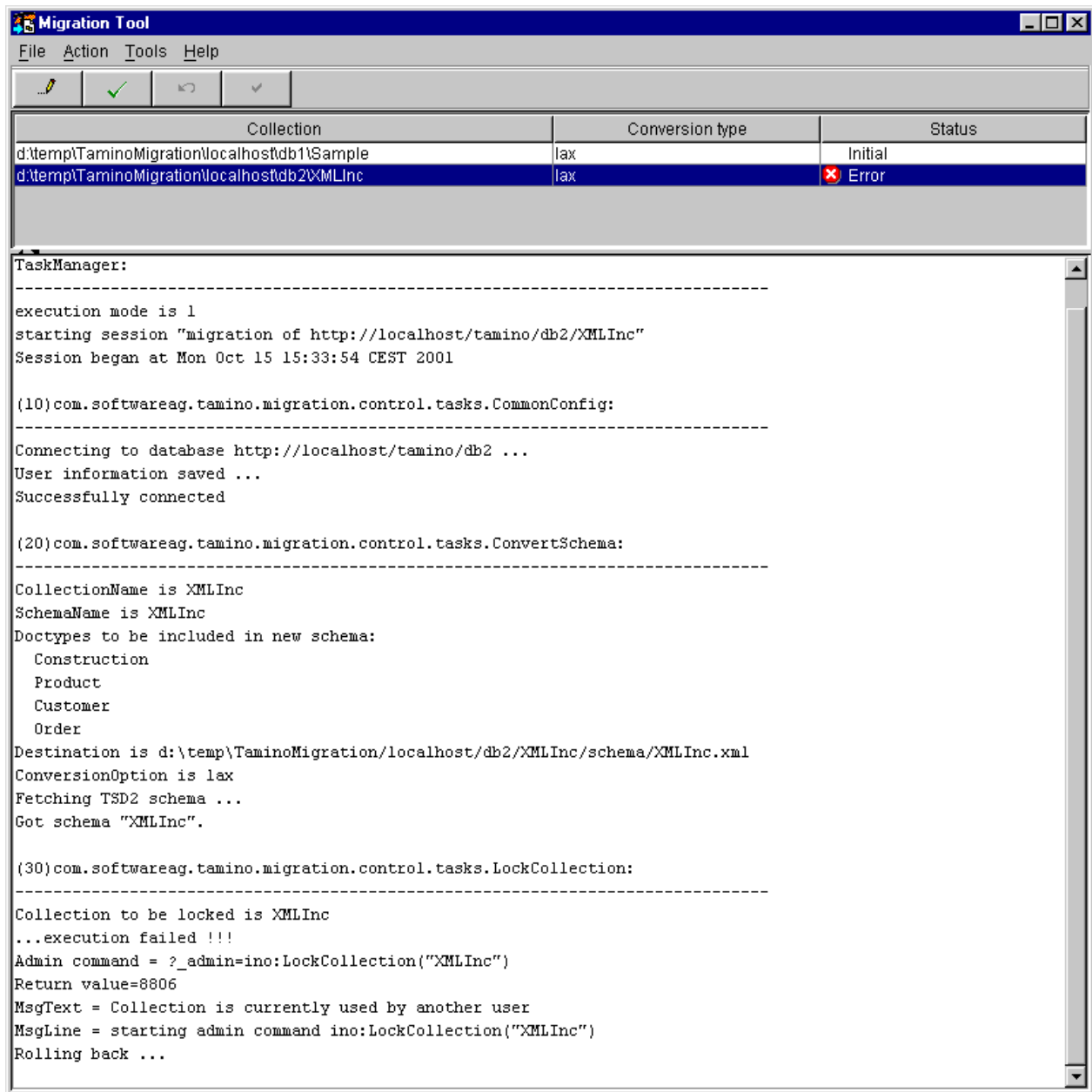
If this message occurs during the execution of step (30) in module **com.softwareag.tamino.migration.control.tasks.LockCollection** during the administration command 'LockCollection' this indicates that the database is correctly started but the version number is not set correctly. Set the version number of the database correctly to the value 3.1 using the Tamino Manager.

ERROR: The database is not in version 3.



As this is easy to recognize, this error situation also occurs in cases when no 'Set Version' operation has been performed in *Tamino Manager* prior to starting the migration tool. The migration tool requires that the version has been set to 3.1.

Collection is currently used by another user. Return value=8806



This message indicates that your database is locked, for example if another user accesses your database or if you started two instances of the Tamino migration tool.

Remedy:

Unlock the database using the `Reset State` command.