



## Reference Guide

Valid from Transit/TermStar NXT Service Pack 16  
2023-11



Transit/TermStar

---

Valid from Service Pack 16 Revised 2023-11. This document is valid as of Transit/TermStar NXT Service Pack 16. Transit/TermStar is being continuously further developed. You can find current Service Packs, installation instructions, user documentation and accessories on our website in the » [Downloads](#) | [Transit & TermStar](#) area.

Contact STAR Group is represented globally in over 30 countries. You can find your local STAR subsidiary on our website under » [Company](#) | [STAR Group worldwide](#).

STAR Group Headquarters:

STAR AG

Wiesholz 35

8262 Ramsen

Switzerland

[www.star-group.net](http://www.star-group.net)

[info@star-group.net](mailto:info@star-group.net)

STAR Language Technology & Solutions GmbH

Umberto-Nobile-Straße 19

71063 Sindelfingen

Germany

Copyright, liability and trademarks Copyright STAR AG

All parts of this documentation are protected by copyright.

Any use outside the limits of copyright law is not permitted without the written consent of the publisher. This applies in particular to the duplication, distribution and translation of this documentation or parts thereof as well as to the storage and processing of the content with electronic data processing systems.

The content of this document has been carefully checked. STAR AG cannot be held liable for any consequences arising from the use of this documentation.

The trademarks used within this document are the property of their respective rights holders.

# Contents

<b>1</b>	<b>Exchanging reference material via TMX</b>	
	What you should know here .....	9
	Supported TMX versions.....	9
	Importing TMX files into Transit.....	10
	Naming and saving the language pairs.....	10
	Filename extension when importing from other systems .....	10
	Performing a TMX import .....	10
	Segment status of imported TMX files .....	11
	Exporting language pairs as TMX file.....	12
	Checking prior to the TMX export.....	12
	Settings for the TMX export.....	12
	Exporting the current project as TMX file.....	13
	Exporting projects, folders, and language files as TMX file.....	14
	Compatibility of the language and country codes of TMX files .....	15
	What you should know here .....	15
	TMX files from other systems in Transit.....	15
	TMX files from Transit in SDL Trados.....	15
<b>2</b>	<b>Managing roles</b>	
	What you should know here.....	17
	Hierarchy of functions.....	17
	Opening a role .....	18
	Creating a new role .....	20
	Creating completely new roles.....	20
	Editing an existing role .....	21
	Protect role by password .....	22
<b>3</b>	<b>Automating tasks using macros</b>	
	What you should know here.....	24
	Tips for creating macros .....	24
	Recording a new macro .....	25
	Running the macro.....	27
	Run a macro via hotkey .....	27
	Run a macro via ribbon bar .....	27

Editing the macro .....	28
Display and edit a macro.....	28
Delete a macro .....	28
Example macro: Inserting Unicode characters.....	29
<b>4 Print page setup</b>	
What you should know here.....	31
Managing print page setups.....	31
Opening a print page setup .....	31
Saving the print page setup .....	32
Print page settings .....	33
Page margins .....	33
Separators for dictionaries.....	34
Headers and footers .....	36
Page layout .....	38
<b>5 Customising dictionary layouts</b>	
What you should know here .....	39
Managing layouts .....	40
Creating a new layout .....	40
Opening existing layouts .....	40
Saving the layout and closing the layout editor .....	41
Providing other TermStar users with your layouts .....	42
Working with the layout editor .....	43
Areas of a layout .....	43
Layout editor interface.....	43
Available fields.....	43
Structure of the language units .....	44
Editing the layout .....	45
Selecting and removing fields for the layout.....	45
Adding and deleting a static text.....	46
Defining the order of fields .....	47
Defining field properties .....	48
What you should know here .....	48
Formatting header and language units.....	48
Formatting language entries and subentries.....	49
Formatting fields .....	50
Entering and formatting static texts .....	51
Variables.....	51
Formatting automatic cross-references.....	51

---

General layout settings.....	53
<b>6 Using start parameters</b>	
Specifying the dialog language: -DialogLanguage .....	55
Specifying the user: -U .....	56
Specifying the user role: -H .....	56
Open project: -P .....	57
Open language pair: -o .....	57
Example using all start parameters .....	57
<b>7 Managing database links</b>	
Transferring database connections to other computers.....	58
What you should know here.....	58
Save ODBC settings to file.....	59
Set up ODBC settings on the target computer .....	59
Finding and deleting incorrect database links .....	61
<b>8 Customising Transit / TermStar</b>	
Segmentation by sentence: Correcting abbreviation lists.....	63
What you should know here .....	63
Filenames of abbreviation lists.....	63
Scopes and storage locations .....	63
Interaction of ewl files and scopes.....	64
Spellchecking: Correcting a list of unknown words .....	65
What you should know here.....	65
Filenames and storage locations.....	65
TermStar: Customising index buttons.....	66
What you should know here.....	66
Configuration files.....	66
Content of configuration files.....	66
<b>9 Organising reference material</b>	
Copying, moving and deleting reference material.....	68
What you should know here.....	68
Three steps of the function .....	68
"All associated language files.." message .....	69
Using the "Organise reference material" function.....	70
Saving and loading the scan results.....	74

Modifying reference material .....	75
Search options and filter criteria .....	75
Exact matches and fuzzy/morpho matches .....	76
"Do not permit as reference material" or delete segment content? .....	77
Using the "Modify reference material" function .....	78
Saving result lists .....	80
Compacting reference material .....	81
Size of the compacted files .....	81
Compacting options .....	81
Compacting the reference material .....	82
10	<b>Open source spellcheck dictionaries</b>
What you should know here .....	84
Filenames of aff and dic files .....	84
Installing an open source spellcheck dictionary .....	85
Uninstalling an open source spellcheck dictionary .....	85
11	<b>Transferring TermStar databases from Access to Microsoft SQL Server</b>
What you should know here .....	86
Renaming the existing ODBC connection .....	87
Creating the new SQL database .....	90
Transferring the database to the new SQL server .....	95
Deleting the connection to the Access database .....	97
Windows 7: Starting the 32-bit ODBC Data Source Administrator .....	98
12	<b>Fields in the TermStar dictionary</b>
What you should know here .....	101
Field formats .....	101
Header fields .....	102
Language fields .....	103
Language entry fields and subentry fields .....	104
Prefixes for field types .....	105
13	<b>Regular expressions</b>
What are regular expressions? .....	107
Basic settings for searches in Transit .....	107
What can you use regular expressions for? .....	108

Overview of meta and control characters .....	109
Defining regular expressions .....	111
Control characters .....	112
Overview of meta characters .....	113
Wildcards: . [ ] & .....	114
Wildcard for any single character: . (dot) .....	114
Wildcard for any of a specified group or class: [ ] .....	114
Wildcard for any sequence of characters: & .....	116
Quantifiers: + * ? .....	118
Escapement: \ .....	121
Applying meta characters to character strings: ( ) .....	123
Placement: ^ \$ .....	124
Negation: ! .....	126
Negation of a character or character string .....	126
Negation of a character group .....	127
Negation of beginning/end of line .....	128
Alternatives:   .....	129
What does Transit interpret as an alternative? .....	129
Alternatives and character groups/classes .....	130
Alternatives and negated character strings .....	130
Variables: # .....	132
Why use variables? .....	132
How are variables used? .....	133
Changing the case when replacing .....	134
Changing number formats when replacing .....	135
Performing mathematical calculations when replacing .....	136
Rounding figures when replacing .....	138
Converting numbers to characters when replacing, and vice versa .....	139
Invalid regular expressions .....	140
Ambiguous regular expressions .....	140
Syntax errors .....	140

## 14 Supported working languages

Sorted by language name .....	142
Sorted by language code .....	148





# 1 Exchanging reference material via TMX

**What you should know here** TMX is a data format for exchanging translation memories. It allows you to use the translation memory from another system in Transit.

You have the following options:

- Importing TMX files into Transit (» [page 10](#))
- Exporting the current project as TMX file (» [page 13](#))
- Exporting projects, folders, and language files as TMX file (» [page 14](#))

**Supported TMX versions** For import, Transit supports TMX versions 1.1 to 1.4.  
For export, Transit uses TMX version 1.4.

## Importing TMX files into Transit



### Importing TMX files into the Transit TM Container

You can also import TMX files from other translation memory systems directly into the optional TM Container (» [Document "Transit: Managing and Using TM Containers"](#)).

Naming and saving the language pairs

When you import TMX files into Transit, Transit generates language files:

- When you import a TMX file from another translation memory system, Transit creates one language pair. You specify the name of the language pair during the import.

If the TMX file contains over 15,000 segments, Transit automatically splits the data into multiple language pairs. A suffix is added to the language pair names.

- When you import a TMX file that has been created using Transit, Transit re-establishes the original language pairs and the original filenames.

If your TMX material has been created from multiple reference files, Transit creates several language pairs that each have the original filename. This means you can work – as usual – with single files and select them individually as reference material.

Filename extension when importing from other systems

When you import TMX files from other translation memory systems, you will need to specify a filename for the language files that you want Transit to create. Specify `.txt` as the filename extension.

Transit then names the language files using the filename that you specified and the filename extension in accordance with the language code.

- Example:

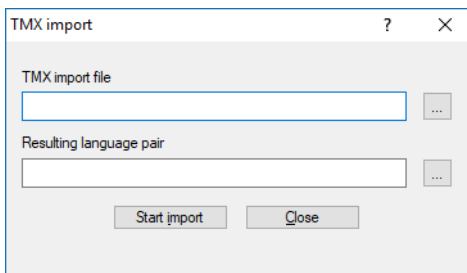
When importing an English-Swedish TMX file, specify the filename `manual.txt`. Transit then creates the language files `manual.eng` and `manual.sve`.

Performing a TMX import

### How do I import a TMX file into Transit?

1. Select **Reference material | TMX interface | Import TMX** from the resource bar.

Transit displays the following window:



2. Select the TMX file you wish to import.
  - Click ... to the right of the **TMX import file** field.  
Transit displays the **Open TMX file** window.
  - Select the TMX file. Confirm your selection with **Open**.
3. Specify the folder and the filenames for the language files which Transit should generate.
  - Click ... to the right of the **Resulting language pair** field.  
Transit displays the **Create language pairs from TMX file** window.
  - Specify the folder and name for the language file. Enter .txt as the filename extension (» **Filename extension when importing from other systems**, page 10).
  - If the TMX file has been created by Transit, the filename that you specify is irrelevant (» **Naming and saving the language pairs**, page 10).
  - Confirm your entries with **Save**.
4. Click **Start**.  
Once Transit has completed the import, it displays the following message:  
Completed successfully.  
Transit has imported the TMX file and saved it as language pairs.
5. Close the **Import progress** window with **OK**.  
Transit displays the **TMX Import** window again in the foreground.  
If you do not want to carry out another import, simply click **Close**.

Segment status of imported TMX files	<p>During the TMX import, Transit assigns the segment status <code>Translated</code> to the source and target languages of the language pairs created.</p> <p>Depending on the quality of the translation memory you imported, you may need to carry out a quality check before using the language pairs created as reference material in translation projects.</p> <p>You can do so either by checking each segment and assigning it a status individually or by changing the status globally for the entire language file (» <b>Transit User Guide</b>, section "Proofreading mode").</p>
--------------------------------------	---

## Exporting language pairs as TMX file



### Exporting the translation memory from the TM Container as a TMX file

You can also export the translation memory from the optional TM Container as a TMX file (» [Document "Transit: Managing and Using TM Containers"](#)).

#### Checking prior to the TMX export

For the export as TMX, you must know the following:

- Whether protected and unaltered segments should also be exported.
- Whether the TMX file should be coded as UTF-8 or UTF-16.

If you want to make the TMX file available to third parties, clarify with these parties how they require the TMX file.

#### Settings for the TMX export

You can define the following for the export:

- **Source language and export languages**

For the export, you must define the source language and at least one export language. The source language is always exported, regardless of whether you select it as an export language.

- **Also export protected segments**

This setting is only required if you want to import the TMX file into Transit at a later point. It is not relevant for exchanging with other systems.

This is used to specify that Transit exports segments that only contain markups.

Otherwise, Transit only exports segments whose content can be edited and translated.

If you select this option, you must also select **Also export unaltered segments**.

- **Also export unaltered segments**

This is used to specify that Transit exports a segment even if its contents (including markups) is identical in the source and target language.

Otherwise, Transit only exports segments whose contents differ between the source language and the target language.

- **Minimum segment status**

This is used to specify that Transit only exports segments that have at least the defined segment status.

Otherwise, Transit exports the segments regardless of their segment status.

- **Export identical segments once only**

This is used to specify that Transit exports identical segments ("*internal repetitions*") only once if they have been translated identically. If identical source language segments have different translations, Transit exports all translation variants.

## Coding the TMX file

This is used to define which coding should be used for the TMX file (UTF-8 or UTF-16).

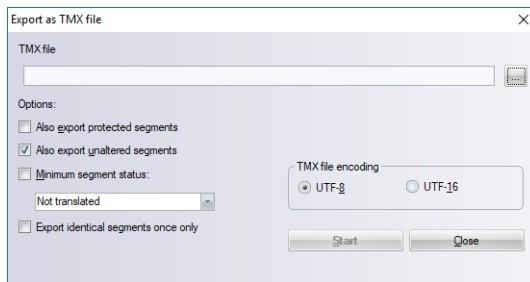
Exporting the current project as TMX file

You can export the language pairs for the current project as TMX file. Transit then exports the source language and all target languages of the project.

### How do I export a project as TMX file?

1. Select **Reference material | TMX interface | Export current project as TMX** from the resource bar.

Transit displays the following window:



2. Specify the TMX file to which the data should be exported:
  - Click ....
  - Select the folder and enter a filename.
  - Confirm your selection with **Save**.
3. Define the additional TMX options and select the TMX file encoding (» [Settings for the TMX export](#), page 12).
4. Click **Start**.

Once Transit has completed the export, it displays the following message: **Completed successfully.**

Transit has created the TMX file.

Close the **Export progress** window with **OK**.

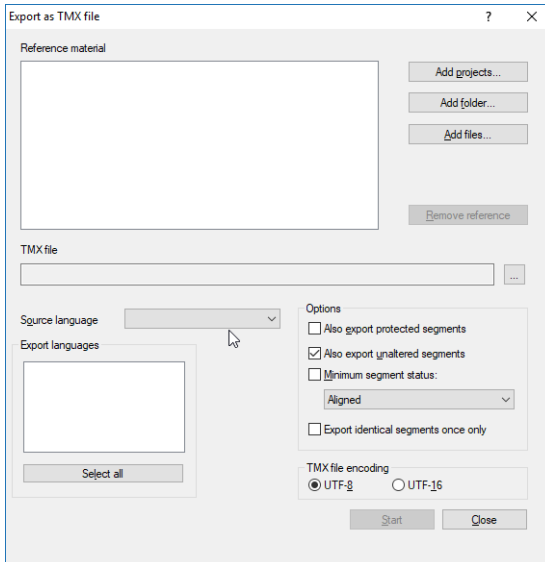
Exporting projects, folders, and language files as TMX file

You can export any project, reference folder, and language file as TMX file.

### How do I export projects, reference folders, or language files as TMX file?

1. Select **Reference material | TMX interface | Export TMX** from the resource bar.

Transit displays the following window:



2. Select the reference material that you want to export.
  - Language files for a project: Select **Add project** and select the project.  
Decide whether you also want to export the reference material of the selected project.
  - All of the language files in a folder: Select **Add folder** and select the folder.
  - Individual language file: Select **Add files** and select the file.You can also select multiple projects, folders and language files to export them together.
3. Specify the TMX file to which the data should be exported:
  - Click ....
  - Select the folder and enter a filename.
  - Confirm your selection with **Save**.
4. Define the source and export languages (» [Settings for the TMX export](#), page 12).
5. Define the additional TMX options and select the TMX file encoding (» [Settings for the TMX export](#), page 12).

6. Click **Start**.

Once Transit has completed the export, it displays the following message:  
Completed successfully.

Transit has created the TMX file.

Close the **Export progress** window with **OK**.

## Compatibility of the language and country codes of TMX files

What you should know here

TMX uses a combination of language and country codes to label languages in accordance with ISO 639-1 and ISO 3166.

For some language variants, different translation memory systems do not use the same codes. This means that some language variants will not be recognised if they have been exchanged between different translation memory systems.

TMX files from other systems in Transit

During the import, Transit takes varying codes from other systems into consideration and can usually correctly interpret and import them.

Exception: The language variants of Serbian (Cyrillic and Latin for Serbia, for Montenegro and for Bosnia and Herzegovina in each case) cannot be clearly assigned.



### Tip: Renaming the language file after the import

If an "incorrect" language variant is created during the import, you can easily rename the filename extension for the language file.

Example:

During the TMX import, a language is interpreted as Serbian (Cyrillic, Montenegro) (language code SCM), but it should be Serbian (Cyrillic, Bosnia and Herzegovina) (language code SRC).

You can correct this by changing the filename extensions for the language files from \*.SCM to \*.SRC.

TMX files from Transit in SDL Trados

SDL Trados does not import the following languages correctly:

Language	Language and country codes		
	According to ISO	Export from Transit	Code in SDL Trados
Afrikaans	af	af	af-01
Basque	eu	eu	eu-01
Catalan	ca	ca	ca-01
Farsi	fa	fa	fa-01
Hebrew	he-il	iw-il	iw-01

Differing coding for special language variants

Language	Language and country codes		
	According to ISO	Export from Transit	Code in SDL Trados
Norwegian (Nynorsk)	nn-no	no-ny	no-ny
Norwegian (Bokmal)	nb-no	no-no	no-ny
Serbian	sr-yu	sr-yu	sh-yu

Differing coding for special language variants



### Tip: Adapting the TMX file before importing

If the translation memory system that is importing the file cannot correctly interpret a language code, it may be helpful to edit the TMX file so that the language codes it contains meet the requirements of the system that is importing the file.



## 2 Managing roles

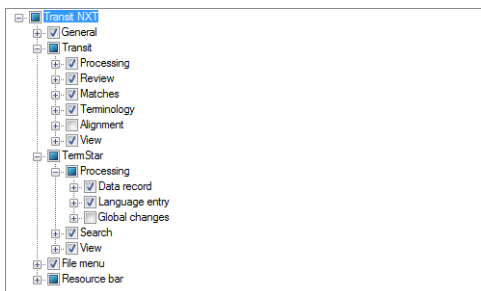
What you should know here Using the role administration of Transit, you can view the standard roles that are supplied (» [Opening a role](#), page 18) as well as create your own roles (» [Creating a new role](#), page 20).

Hierarchy of functions The role administration hierarchically displays the functional elements of a role:

- **General:** General areas of the ribbon bar
- **Transit:** Transit-specific areas of the ribbon bar
- **TermStar:** TermStar-specific areas of the ribbon bar
- **File menu:** Menu and submenus of the Transit button
- **Resource bar:** Buttons and submenus of the resource bar

The check box on the left of an element indicates if the role supports the function:

- **Check mark:** Function and all subfunctions are supported.
- **Empty:** Function and all subfunctions are not supported.
- **Filled:** Function is partially supported (some, but not all subfunctions).



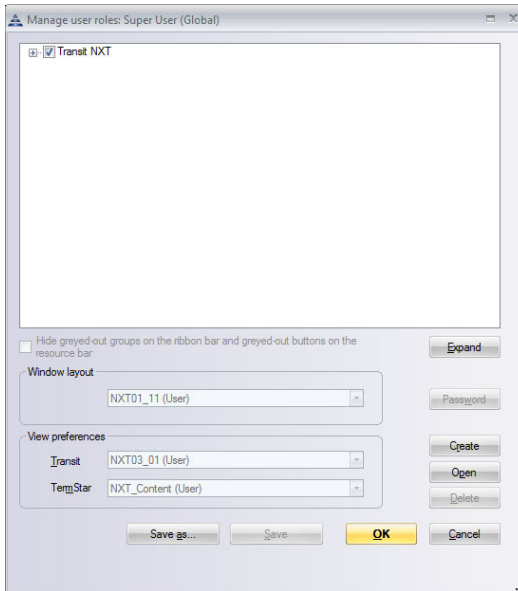
Example: Functions of the Localisation Specialist role

In addition, windows and views are linked to the role.

## Opening a role

### How do I open a user role?

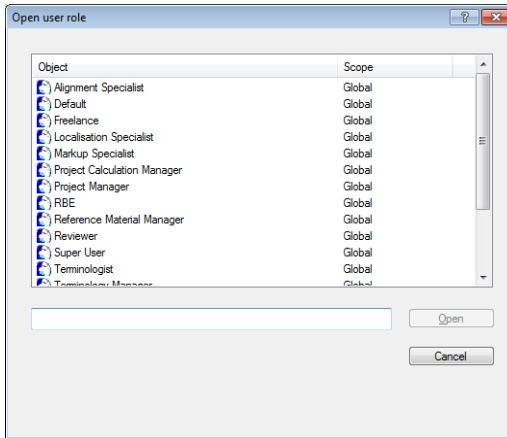
1. Select **User roles | Manage user roles** from the resource bar.  
Transit opens the following window:



Transit displays the settings of the active role.

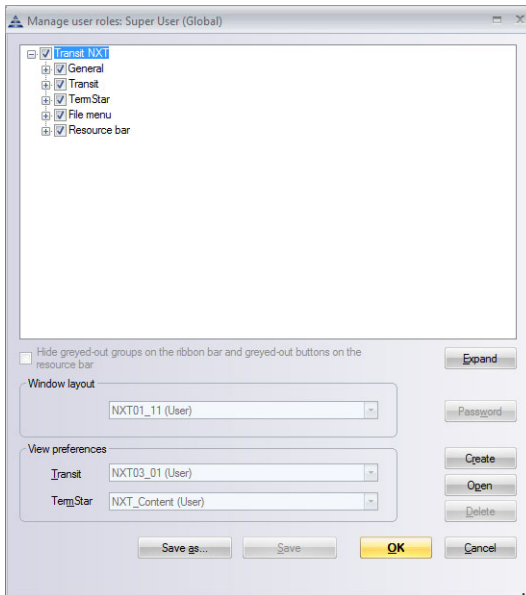
2. Select **Open**.

Transit opens the following window:



3. Select the user role you want to open and click **Open**.

Transit displays the selected role:



Settings of the Super User role

The upper section of the window hierarchically displays the function levels (» [Hierarchy of functions](#), page 17).

- To display or hide the next-lowest level, click the plus sign or minus sign on the left of the name.
- To display all sub-levels, click **Expand**. To hide all sub-levels again, click once again on **Expand**.

In the **Windows** and **View** lists, you will find the windows assignment as well as the Transit and TermStar views of the role.

## Creating a new role

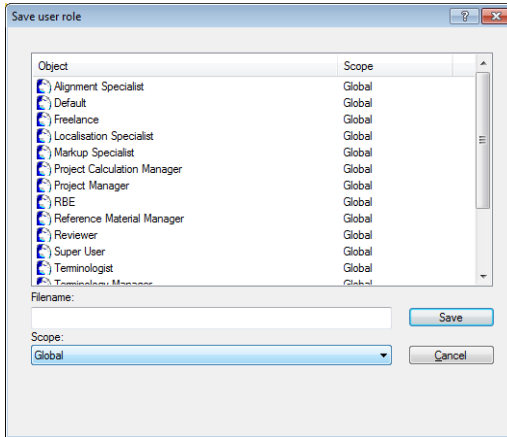
You can create a completely new role. However, it is generally easier to open an existing role, save it under a different name and then edit it (» [Editing an existing role](#), page 21).

Creating completely new roles

### How do I create a completely new role?

1. Select **User roles | Manage user roles** from the resource bar.  
Transit displays the **Manage user roles** window.
2. Click **Create**.  
The upper section of the window hierarchically displays the function levels. At the beginning, all functions and subfunctions are checked, i.e. are supported by the role.
3. Restrict the functions you want to be available for the role.  
To do this, deselect the relevant functions and sub-functions.
4. Decide how inactive controls (groups in the ribbon bar or buttons in the resource bar) are displayed:
  - To completely hide these controls, select the option **Hide greyed-out groups on the ribbon bar and greyed-out buttons on the resource bar**.  
In this case, the user will not see the elements.
  - If you do not select this option, the controls will be greyed out. The user will be able to see these controls but not use them.
5. In the **Windows** area, select the window layout for the role.
6. In the **View preferences** area, select the Transit and TermStar views for the role.
7. Click **Save**.

Transit displays the following window:



Transit displays the existing roles.

8. Enter a name for the new role in the **Filename** field.
9. Under **Scope**, select the scope for the role:
  - global: For all users and all projects
  - user: Only for the current user

Click **Save** to confirm the information entered.

Transit saves the role under the selected name. As an option, a password can be allocated for the role (» [Protect role by password](#), page 22).

### Editing an existing role **How do I edit an existing role?**

1. Open the role (» [Opening a role](#), page 18).
2. Click **Save As**.

Transit displays the **Save user role** window with the existing roles.

- Enter a name for the new role in the **Filename** field.
- Under **Scope**, select the scope for the role.

Click **Save** to save the information entered.

Transit saves the role under the selected name.

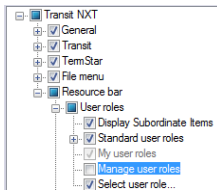
3. Edit the role settings (details » [Creating a new role](#), page 20):
  - Define the functions that you want to be available for the role.
  - Decide how inactive controls are displayed (option **Hide greyed-out groups on the ribbon bar and greyed-out buttons on the resource bar**).
  - In the **Windows** area, select the window layout for the role.

- In the **View preferences** area, select the Transit and TermStar views for the role.
4. As an option, a password can be allocated for the role
  5. Click **Save** to save the information entered.
- Click **OK** to use the new role or **Cancel** to go back to the current role.

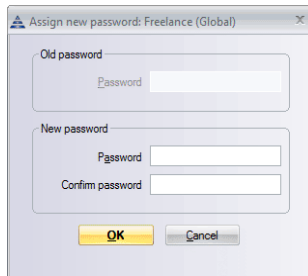
## Protect role by password

You can protect user roles with passwords. Doing so, a particular user is only able to work with the specified role and cannot change it.

1. Specify the role settings (» [Creating completely new roles](#), page 20 or » [Editing an existing role](#), page 21).
2. To ensure that the user cannot change the role deactivate the **Managing roles** option:



3. Click **Password**.  
Transit displays the following window:



You must enter the new password in exactly the same way in both fields.

Enter the new password, enter it a second time to confirm, and then select **OK** to confirm your entry.

4. Ensure that the user cannot select a standard role.  
To do so, add the following parameter to the [options] section in the `starte.ini` file:  
`StdActorsDisabled=1`

To ensure this setting cannot be changed, the user must not have access to the `\bin` folder of his or her Transit installation.

If the user has already set up individual roles, you must delete them.

5. Send the ACT file for the role and the relevant password to the user.

You will find the ACT file in the subfolder of the selected scope in the `\config` folder.

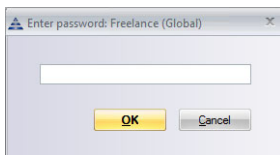
### How do I use a password-protected role as a user?

1. Place the ACT file for the role in the subfolder for the desired section in the `\config` folder.

2. If Transit is already open, the role can be opened (» [Opening a role](#), page 18).

If you are restarting Transit, you can open the role quicker by selecting it directly in the **Role overview** window or via the resource bar (**Roles | My roles**).

Transit displays the following window:



3. Enter the password.

# 3 Automating tasks using macros

**What you should know here** You can use a macro to automate a task you want to perform repeatedly in Transit. For this purpose, you record the necessary steps of the task in a macro (» [Recording a new macro](#), page 25). Transit saves all the commands and inputs you give when recording the macro. Then you can run the macro again and again. In this way (» [Running the macro](#), page 27), Transit automatically carries out all the commands and inputs contained in the macro. You can also alter, correct or delete an existing macro (» [Editing the macro](#), page 28).

- Tips for creating macros**
- Think one step ahead  
Before recording a macro, plan out the steps and the commands which you want the macro to perform.  
If you make and correct an error while recording a macro, the macro will record the error as well as its correction.  
You can edit the macro later and remove unnecessary steps – but it is easier to record it correctly.
  - Avoid unnecessary queries and messages  
Example:  
If you close a modified language file, Transit displays a message if you have not yet saved the file. To avoid Transit displaying the message in the middle of a macro, save the language file before using the **Close language pair** function.
  - Independent from content  
If you wish to use the macro recorded for other language files, ensure that the macro is not dependent upon the contents of the current language file.
  - Select the keyboard shortcut  
For each macro, you specify a keyboard shortcut which you can use to start the macro.  
Ensure that you do not use standard Transit keyboard shortcuts with which you normally work.



- Do not use mouse movements  
Transit does not record any mouse movements in macros, but only actions and steps you carry via keyboard.  
For this reason, only use the keyboard and call-up menus and commands only with keyboard shortcuts.  
If you are not familiar with keyboard shortcuts, note them before recording the macro.

## Recording a new macro

Transit saves all the steps you give when recording a macro.



### Recording macros without mouse movements

Only use keyboard shortcuts or keyboard input for macros.

Transit does not record any mouse movements. This means that steps you carry out using the mouse are lost.

### How do I record a macro?

1. Select **Edit | Macros | Record**.

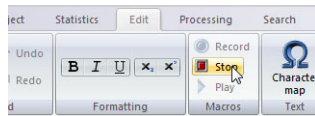
Transit displays the following window:

2. In the **Macro name** field, enter a name for the macro.
3. In the **Hotkey** section, specify the hotkey with which you want to run the macro.
  - Select **SHIFT+** for a keyboard shortcut using the “Shift” key.
  - Select **CTRL+** for a keyboard shortcut using the “Control” key.
  - From the **Key** list, select the key with which you want to run the macro – if necessary, in conjunction with the SHIFT and/or CTRL key.
4. To have Transit display the commands recorded after recording the macro, select **Show macro definition after recording**.

Transit can also display the macro definition any time at a later date (» [Editing the macro](#), page 28).

5. Confirm your entry with **OK**.
  - Macro name already exists message:  
Confirm the message with **OK** and in the **Macro name** field, enter another name which is not already in use (» [step 2](#), page 25).
  - No hotkey selected message:  
Confirm the message with **OK** and select a key from the **Key list** (» [step 3](#), page 25).

From this point, Transit records all the commands and inputs you execute using the keyboard.
6. Carry out all the commands and keyboard inputs which you want to record in the macro.
7. To stop recording the macro, use the mouse to select **Edit | Macros | Stop**.  
Make sure you select this option using the mouse. Do not use a keyboard shortcut as this would be recorded with the macro.



Stop the recording using the mouse, but not using the keyboard.

Transit stops recording the macro.

If you selected the **Show macro definition after recording** option (» [step 4](#), page 25), Transit displays recorded macro (» [Editing the macro](#), page 28).

## Running the macro

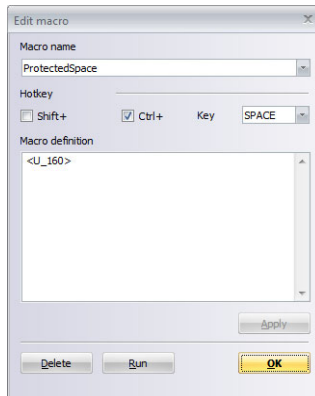
When you run a recorded macro, Transit carries out all the steps saved in the macro.

### Run a macro via hotkey **How do I run a macro using a hotkey?**

1. Press the hotkey which you specified for the macro.  
You specified this hotkey while recording the macro (» [step 3](#), page 25).  
Transit runs the macro selected.

### Run a macro via ribbon bar **How do I run a macro using the ribbon bar?**

1. Select **Edit | Macros | Play**.  
Transit displays the following window:



2. From the **Macro name** list, select the macro which you want to run.
3. Click **Run**.

Transit runs the macro selected.

## Editing the macro

You can edit and delete existing macros.



### Re-recording an existing macro

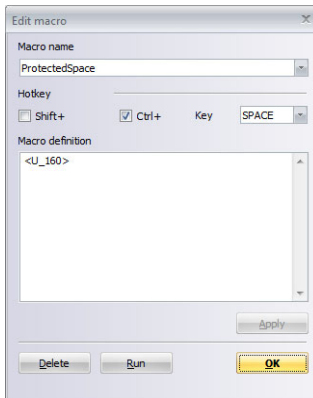
If you want to re-record an existing macro, delete the macro (» [Delete a macro](#), page 28) and then record it under the same name (» [Recording a new macro](#), page 25).

Display and edit a macro

### How do I display and edit a macro?

1. Select **Edit | Macros | Play**.

Transit displays the following window:



2. From the **Macro name** list, select the macro which Transit should display. Transit displays the shortcut and the contents of the macro.
3. Change the macro settings:
  - To change the shortcut, adapt the settings in the **Hotkey** section.
  - To edit the recorded commands, correct them in the **Macro definition** section.
4. Click **Apply** and close the window with **OK**.

Delete a macro

### How do I delete a macro?

1. Select **Edit | Macros | Play**.

Transit displays the **Edit macro** window.

2. From the **Macro name** list, select the macro which you want to delete.
3. Click **Delete** and close the window with **OK**.

Transit deletes the macro.

## Example macro: Inserting Unicode characters

To insert Unicode characters, you can click **Edit | Text | Character map** to open the **Character map** window.

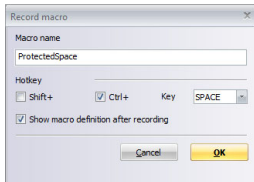
However, if you need a Unicode character very often, there is an easier way: You specify a macro, e.g. for a non-breaking space (Unicode character 160).

### How do I record a macro used for inserting a Unicode character?

1. Select **Edit | Macros | Record**.

Transit displays the **Record macro** window.

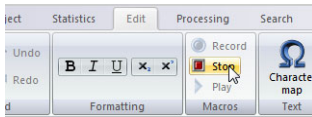
- In the **Macro name** field, enter a name for the macro.  
Example: Non-breaking space
- In the **Hotkey** section, specify the hotkey with which you want to run the macro.  
Example: **Ctrl + SPACE**
- Select **Show macro definition after recording**.



Confirm your entries with **OK**.

From this point, Transit records all the keyboard input and commands you execute using keyboard shortcuts.

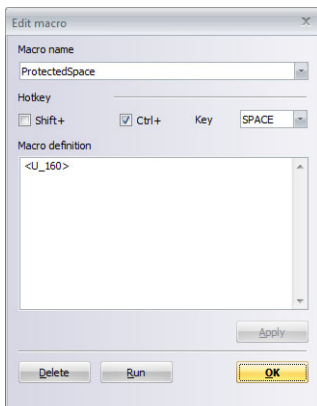
2. Finish recording of the macro immediately afterwards. To do so, select **Edit | Macros | Stop** using the mouse.



Stop the recording using the mouse, but not using the keyboard.

Transit displays the empty macro in the **Edit macro** window.

3. In the **Macro definition** field, enter the command that Transit uses for inserting the Unicode character in text:



The command `<U_160>` inserts the Unicode character 160 (non-breaking space)

4. To confirm the changes, click **Apply** and close the window with **OK**.

Now, you can simply insert the Unicode character in Transit with the defined shortcut: For a non-breaking space you press CTRL + Space bar.

# 4 Print page setup

What you should know here When printing language pairs or dictionaries, you can save several settings as a print page setup and use it again at a later stage (» [Opening a print page setup](#), page 31 and » [Saving the print page setup](#), page 32).

The print page setup can be used for the following settings:

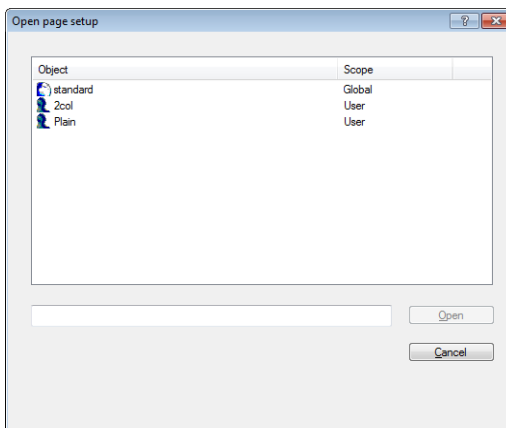
- Page margins (» [page 33](#))
- Separators for dictionaries (» [page 34](#))
- Headers and footers (» [page 36](#))
- Page layout (» [page 38](#))

## Managing print page setups

Opening a print page setup **How do I open an existing print page setup?**

1. Select **Transit button | Print | Page setup**.
2. Click **Open**.

Transit displays the following window:



3. Select the print page setup and click **Open**.
- Transit displays the settings for the print page setup opened.

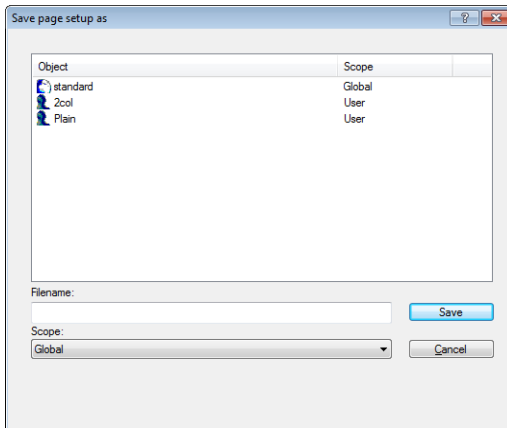
**Saving the print page setup** If you have changed a print page setup, you can save it with the new settings. You have two options here:

- **Save:** Save as an existing print page setup  
Transit saves the settings in the opened print page setup and overwrites the old settings.  
To do so, click **Save** in the **Page setup** window.
- **Save as:** Save as a new print page setup  
With this function, you create a new print page setup with the new settings. The existing print page setup remains unchanged.

### How do I save the changes as a new print page setup?

1. In the **Page setup** window, click **Save as**.

Transit displays the following window:



Use **Filename** to specify the name under which you will be able to select the page setup subsequently. You should therefore use descriptive names for this.

2. Enter a name for the new print page setup in the **Filename** field.
3. Under **Scope**, select the scope for the print page setup:
  - **Global:** For all users and all projects
  - **User:** Only for the current user
  - **Customer:** Only for projects of the current customer
4. Click **Save**.



## Print page settings

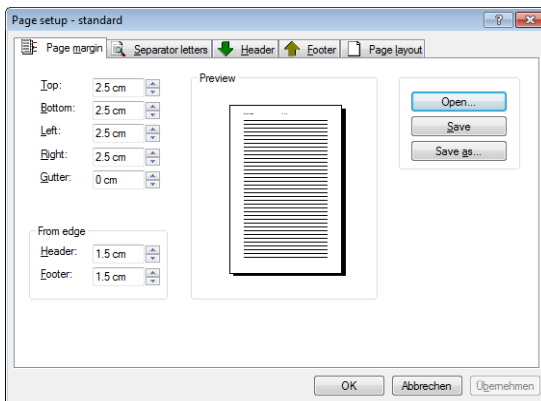
Page margins On the **Page margin** tab, you can set the following settings:

- **Top, Bottom, Left, Right:** Margins on all four sides
- **Gutter:**
  - For one-sided printing: Additional left-hand margin
  - For double-sided printing (» [Page layout](#), page 38): Additional margin along the inside for binding

The gutter applies in addition to the **Right** or **Left** margins.
- **Header:** Distance of header from the top edge  
Ensure that the header margin is smaller than the top page margin. Otherwise the header extends into the print area.
- **Footer:** Distance of footer from the bottom edge  
Ensure that the footer margin is smaller than the bottom page margin. Otherwise the footer extends into the print area.

### How do I set the margins?

1. In the **Page setup** window, select the **Page margin** tab:



2. Specify the values for the margins.

If Transit displays a message when you enter the values, this means that the value is not suitable for the current printer:

- **Minimum value message:** The margin is too small because the printer cannot print right out to the edge of the paper.
- **Maximum value message:** The margin is too big because the margins are larger than the paper format.

Do not forget to save the print page setup if you have changed it (» [Saving the print page setup](#), page 32).

Separators for dictionaries On the **Separator letters** tab, you can specify how TermStar should separate the different letter groups of the dictionary.

You can specify separators for letter ranges, individual letters, digits or a combination of these options.

TermStar distinguishes between upper and lower case. Therefore enter letters and letter ranges in both versions (upper and lower case).

TermStar does not print a separator for letters or digits you do not specify.

Separator type	Entry in separator field (examples)	Meaning
Letter range	a-zA-Z	TermStar separates any letter from A to Z. TermStar prints all other characters without separator (e.g. special characters or numbers).
	d-kD-K	TermStar separates any letter from D to K. TermStar prints all other characters without separator (i.e. letters A, B, C and from L to Z).
Individual letters	aAbBcCâÄ	TermStar separates the specified letters A, B and C as well as Ä. TermStar prints all other characters without separator.
Numbers	1-9	TermStar separates any number from 1 to 9. TermStar prints all other characters without separator (i.e. all letters).
Combinations	1-5a-tA-TâÄ	TermStar separates numbers from 1 to 5, letters from A to T and Ä.

Separators for printing the dictionary

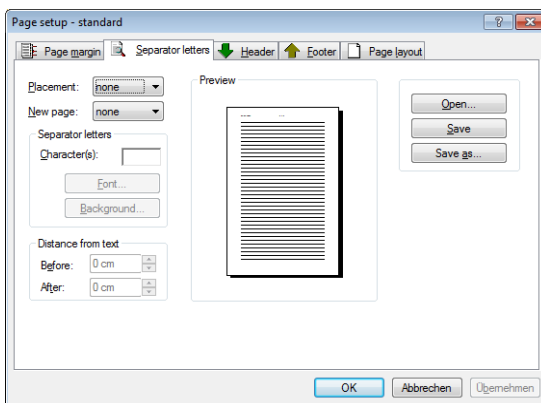


#### No separator letters for German umlauts

German umlauts (Ä, Ö, Ü) are not sorted as individual letters, but are sorted under A, O and U. For this reason, TermStar does not print separators for these umlauts.

## How do I set the separators?

1. In the **Page setup** window, select the **Separator letters** tab:



2. From the **Placement** list, select where TermStar should position the separator:
  - None: TermStar does not print any separators.
  - Left, Centre, or Right: TermStar prints the separator left-aligned, centred or right-aligned.
3. From the **New page** list, select whether TermStar should print any new letter on a new page:
  - None: New letter on the same page
  - Page: New letter on the next available page
  - Right page: New letter on the next right-hand page (may result in a blank left-hand page)

This option is only relevant to double-sided layouts (» [Page layout](#), page 38).

4. In the **Character(s)** field, enter the characters before which TermStar should print a separator (table » [Separators for printing the dictionary](#), page 34).
5. If you want to change the font for the separator, click **Font**.
6. If you want to select a background for the separator, click **Background**.
7. In the **Distance from text** section, specify the distances before and after the separator.

Do not forget to save the print page setup if you have changed it (» [Saving the print page setup](#), page 32).

**Headers and footers** On the **Header** and **Footer** tabs, you can specify text to be printed on each page. For this purpose, you can use static text (e.g. your department, copyright note) and variables (e.g. date, consecutive page number, filename).

A header or footer has left, center and right areas. You can add text and variables to each of these areas. You can also set a font for each area.

In addition, you can specify whether header and footer are to be separated by a line from the print area.

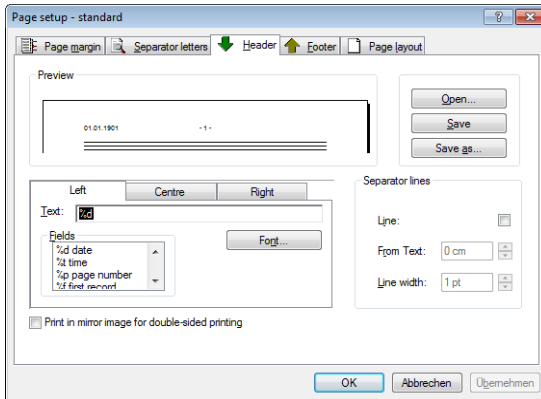
You can use the following variables in headers and footers:

Variable	Meaning	Example
%d	Current date	08.08.2016
%t	Current time	13:39
%p	Page number	12
%f	● Transit: Number of the first segment on the page	126
	● TermStar: Term of the first data record on the page	<i>application</i>
%l	● Transit: Number of the last segment on the page	261
	● TermStar: Term of the last data record on the page	<i>browser</i>
%n	● Transit: Working name of the language file	<i>About Transit NXT</i>
	● Use in TermStar inadvisable	
%N	● Transit: Path and filename of the language file	<i>d:\project\NXTWord\About_Transit NXT</i>
	● TermStar: Dictionary and database name	<i>MyTerms(MyDB)</i>

Variables for header and footer

## How do I specify the header or footer?

1. In the **Page setup** window, select the **Header** or **Footer** tab:



2. Choose the area you want to specify. To do so, select the **Left**, **Center** or **Right** tab.
3. Specify the content for the area selected:
  - To insert a variable, you can double-click the corresponding entry in the **Fields** list.
  - If you want to change the font for this area, click **Font**.
4. Select **Double-sided symmetrical** to print the headers and footers in mirror-image on the right and left pages.

If you have selected this option, the right and left-hand areas will be swapped over between the right and left-hand pages. This means that the page numbers always appear on the outer edge of the page.

The symmetrical layout is only used for double-sided layouts (» [Page layout](#), page 38).

5. You can insert a separating line between the header/footer and the print area of the dictionary:
  - In the **Separator lines** section, check the **Line** option.
  - Specify the distance of the line from the header and footer (**From text** setting) and the **Line width**.

Do not forget to save the print page setup if you have changed it (» [Saving the print page setup](#), page 32).

Page layout On the **Page layout** tab, you can define the general appearance of the printout.

- **Page layout**

- **1st page number**

This is where you specify with which page number the numbering should start.

You can determine **where** the page number is printed using the variable %p in the header or footer (» [Headers and footers](#), page 36).

- **Double-sided**

If you select **Double-sided**, Transit prints left and right pages differently:

The gutter is always added along the inside edge of the page (» [Page margins](#), page 33).

Headers and footers can be printed in mirror-image on the right and left pages (» [Headers and footers](#), page 36, **Double-side symmetrical** option).

- **Column layout**

- **Columns:** Number of columns

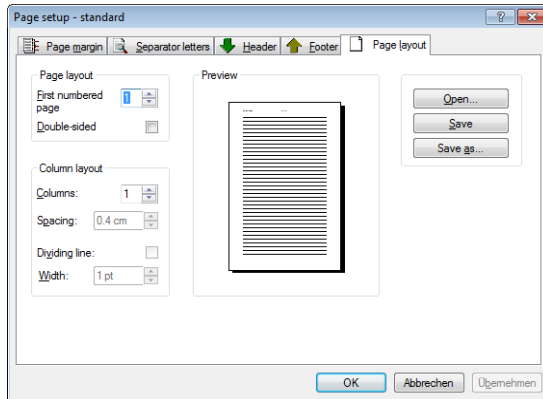
For layouts with more than one column, you can also specify:

- **Spacing:** Spacing between the columns

- **Line between:** TermStar inserts a separating line between the columns. You can also specify the width of the separating line in the **Thickness** field.

### How do I specify the page layout?

1. In the **Page setup** window, select the **Page layout** tab:



2. Specify the page and column layout.

Do not forget to save the print page setup if you have changed it (» [Saving the print page setup](#), page 32).

# 5 Customising dictionary layouts

What you should know here TermStar saves the settings for displaying your dictionaries in views. A view is composed of several layouts. You can use these layouts to define how TermStar displays the fields of dictionaries and which fields can be edited.

The following properties are specified in the layouts:

- Field selection for the header data
- Selecting fields for the language entries (separately for source language, target language and additional languages)
- Field layout
- Formatting the field contents (font, size, colour, etc.)
- Texts that TermStar displays before and after the contents of the field
- Static texts that TermStar displays for every data record
- Display cross-references

TermStar is preconfigured with a number of layouts that have proven successful in practice. You can also create and save your own layouts (» [Managing layouts](#), page 40). For editing layout, you use the *layout editor* (» [Working with the layout editor](#), page 43).

To use a new layout in TermStar, you will need to assign it to one of the existing views (» [TermStar User Guide](#), section "Customising the dictionary window").

# Managing layouts

## Creating a new layout



### Using an existing layout as a template

You can use the layout editor to create a completely new layout. However, it is generally easier to open an existing layout (» [Opening existing layouts](#), page 40), save it under a different name and then edit it.

### How do I create a new layout?

1. Select **View | Terminology layout | Create** and one of the following options:
  - Dictionary layout
  - Bibliography layout
  - Address layout

The layout editor displays a new layout. Now you can edit this layout and then save it (» [Working with the layout editor](#), page 43).

## Opening existing layouts

You have the following options for opening an existing view:

- Open an active layout
- Open any layout

### How do I open an active layout?

1. Select one of the following options depending on the layout you want to edit:
  - Active left layout: **View | Terminology layout | Left | Modify**
  - Active right layout: **View | Terminology layout | Right | Modify**
  - Active editing layout: **View | Terminology layout | Edit | Modify**

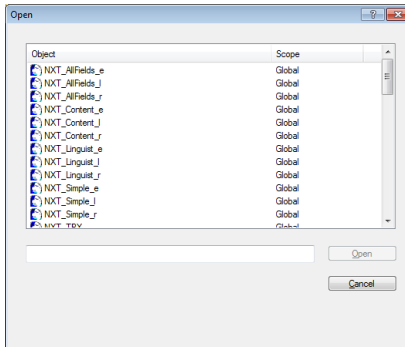
TermStar opens the selected layout in the layout editor (» [Working with the layout editor](#), page 43).

### How do I open any layout?

1. Select **View | Terminology layout | Modify** and one of the following options:
  - Dictionary layout
  - Bibliography layout
  - Address layout



2. TermStar displays the following window:



TermStar displays all of the existing layouts.

3. Select the layout you want to edit and confirm your choice with **Open**.

TermStar opens the layout in the layout editor (» [Working with the layout editor](#), page 43).

### Saving the layout and closing the layout editor

If you have edited a layout in the layout editor, you must save it so that your changes are not lost.

There are two ways of saving a layout:

- **Save:** The layout editor saves the layout under the same name and in doing so overwrites your old layout.
- **Save under a new name:** The layout editor saves the layout under a new name. This allows you to create a new layout without changing your old layout.

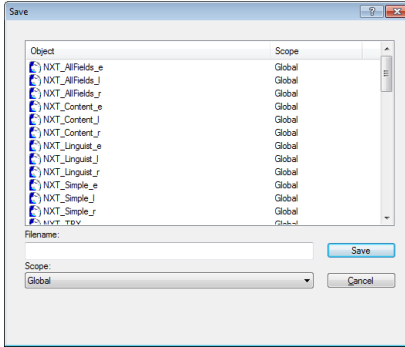
### How do I save a layout?

1. In the header of the layout editor, click **X**.  
Transit displays the following message:  
Layout "... " has been changed. Save?
2. Confirm the message by clicking **Yes**.

**How do I save a layout under another name?**

1. Select **Transit button | Save as**.

TermStar displays the following window:



Use **Filename** to specify the name under which you will be able to select the layout subsequently. You should therefore use a descriptive name.

2. Enter a name for the new layout in the **Filename** field.
3. Select the scope for the new layout from the **Scope** list:
  - `Global`: For all users and all projects
  - `user`: Only for the current user
  - `customer`: Only for projects of the current customer
4. Click **Save** to confirm the information entered.

Transit saves the layout under a new name.

Providing other TermStar users with your layouts

Each layout is saved in its own file with the file extension `tl.d`. You can provide other TermStar users with access to any layout file so that they can also use it as their layout.

Where the layout files are saved depends on the scope that you defined when saving the file (» [How do I save a layout under another name?](#), page 42):

- Global scope: `config\global` folder
- Customer scope: `config\customers\<customer>` folder
- User scope: `config\users\<user>` folder

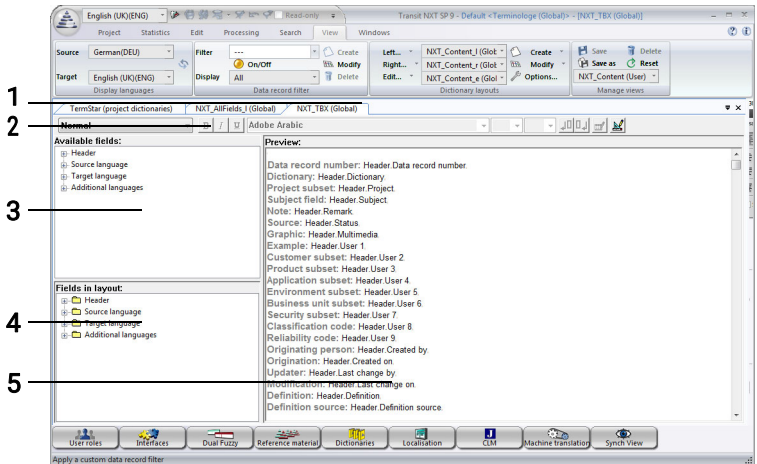
## Working with the layout editor

Areas of a layout A layout consists of the following areas:

- **Normal:** You can use this to define how TermStar displays the main entries and their subentries.
- **Cross-reference:** You can use this to define how TermStar displays the subentries as separate entries in the dictionary and how the automatic cross-reference to the main entry appears.
- **Abbreviation, Alternative, Irregular form, Synonym, Disallowed term, User index 1 to User index 5**

The **Cross-reference** settings apply by default to all subentry types. You can, however, make each subentry type look different.

Layout editor interface When you open or create a layout, TermStar opens the layout editor:



Elements of the layout editor: 1: Tab for layout editor; 2: Toolbar; 3: Available fields (» page 43); 4: Fields used in the layout; 5: Preview

Available fields The layout editor displays all of the fields that are not used in the layout as “available fields”.

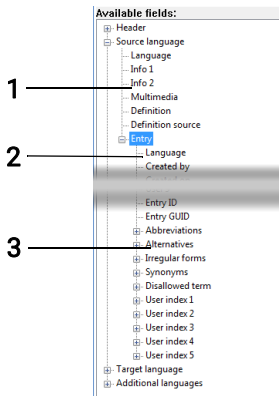
TermStar displays the fields hierarchically:

- **Header:** Header fields which belong to a data record.
- **Source language:** Source language fields.

Example: If you select the **Term** and **Definition** fields here, TermStar displays the source language with these fields.

- **Target language:** Target language fields.  
Example: If you select the **Term** and **Context** fields here, TermStar displays the target language with these fields.
- **Additional languages:** Fields for additional languages  
These fields are relevant for multilingual dictionaries which contain language entries in more than two languages.  
Example: If you select the **Term**, **Context**, **Part of speech** and **Gender** fields here, TermStar displays any languages that are neither the source language nor the target language with these fields.

Structure of the language units The structure of the target, source and additional languages is identical:



Structure of a language unit: 1: Language fields, 2: Language entry fields, 3: Subentry fields

- **Language pseudo-field**  
You can use the **Language** pseudo-field to display the language code or the language name for a language entry.  
This can be selected and formatted like a normal language entry field.  
You can define whether the layout displays the language code or the language name by using a variable in the **Field properties** window (» [Variables](#), page 51).
- **Language fields**  
If, for example, a data record contains multiple language entries in English, you can enter a common definition which applies to all the English language entries of a data record.
- **Language entry fields**  
The **Entry** subgroup of a language unit contains the language entry fields for this language.  
You can specify whether you want TermStar to display multiple language entries for the language unit. Example: If a data record has two English language entries,

TermStar can display only one or both language entries (» [Formatting language entries and subentries](#), page 49).

- Subentry fields

In addition to the fields for the main entry, the subentries and their fields are also available. You will need to add these subentries to the layout in order to display them.

Like in main entries, the `Term` field is the index field whose content is sorted as an own entry in the dictionary.



#### Source language `Term` field must be used

You can only save a layout if it uses the language entry field `Term` in the source language.

Otherwise, TermStar displays the following message:

Layout "... has been changed and is invalid. Continue editing?

In this case, insert the source language `Term` field and try to save the layout again.



#### Automatic cross-references in the "cross-reference" layout area

If you insert the field of a subentry in the normal area of a layout, TermStar displays this subentry within the main entry.

You can also define how TermStar displays subentries as a separate entry with an automatic cross-reference to the main entry (» [Formatting automatic cross-references](#), page 51).

## Editing the layout

Selecting and removing fields for the layout If you want to display and edit fields in the dictionary, the layout must display these fields. Select the fields from the **Available fields** area so that the layout editor displays them in the **Fields in layout** area and in the **Preview**.

TermStar can display each field only once in the layout. Each field is therefore either in the **Fields in layout** area or in the **Available fields** area.

#### How do I add a field to the layout?

1. Select the main layer you require (**header, source language, target language or additional languages**).
  - Click the plus sign on the left of the main layer to make TermStar display its fields or sub-sections.
  - Click the plus sign on the left of the sub-section to make TermStar display its fields.

2. Select the field you want to add to the layout.

If you wish to add all the fields in a layer, click the name of the layer (e.g. **Source language** for all the source language fields).

3. In the context menu, select **Add to layout**.

The layout editor adds this field to the **Fields in layout** area and displays it in the **Preview**.

Please refer to section » [Defining field properties](#), page 48 for information on how to format the field.

#### How do I remove a field from the layout?

1. Select the field from the list in the **Fields in layout** area.

To remove all fields from a layer, select the name of the layer.

2. In the context menu, select **Delete**.

Transit moves the field into the **Available fields** area and removes it from the **Preview**.

Adding and deleting a static text

In addition to fields, you can also add [static text](#). TermStar adds these texts to the **Fields in layout** area. You can move these in the same way as fields (» [Defining the order of fields](#), page 47).



#### Advantages of static texts compared to text before / after a field

You can insert static texts as well as define text that is displayed before or after a field.

Static texts have the following advantages:

- You can move a static text as a separate unit.  
Texts before/after a field are always displayed before or after the field content.
- You can define a separate indent for the static text.  
Texts before / after a field have the same indent as the field content.
- Static texts are always displayed (even if fields are empty).  
If empty fields are not displayed (» [General layout settings](#), page 53), the texts before / after an empty field are not displayed.

#### How do I insert static text?

1. Select a field from the lists in the **Fields in layout** area below which you want to insert static text.
2. In the context menu, select **Insert text**.

The layout editor displays the **Static text** element under the selected field.

Refer to section » [Defining field properties](#), page 48 for information on how to enter and format the text.

### How do I remove static text from the layout?

1. From the list in the **Fields in layout** area, select the static text you want to remove.
2. In the context menu, select **Delete**.

Transit removes the text from the **Fields in layout** and **Preview** areas.

### Defining the order of fields

TermStar arranges the fields in the order in which you add them. You can change the sequence of the fields at a later point by moving the fields in the **Fields in layout** area.



#### Tips for moving fields

- Fields can only be moved within their own layer.

Example: You cannot move any header fields into the source language layer.

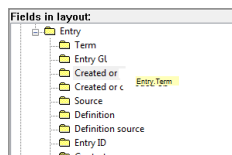
- You can move complete layers.

To do this, move a layer onto another layer of the same hierarchical level (e.g. **Source language** onto **Header** to place it below the header).

### How do I rearrange the order of fields?

1. Select a field from the list in the **Fields in layout** area.
2. Click the field and hold down the mouse button to drag it up or down.

In doing so, the layout editor also moves the field icon:



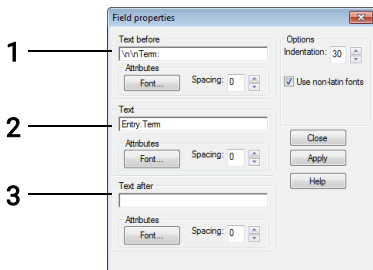
Moving a field in the layout editor

3. Drag the field over another field and release the mouse button.

The layout editor positions the field you moved below the selected field.

## Defining field properties

What you should know here



Field properties in the layout editor: 1: Text before the field contents, 2: Field contents or static text, 3: Text after the field contents

You can leave this window open the whole time so you can format the fields. If you select a field from the **Fields in layout** area, the layout editor displays the formatting of the field in this window.

You have the following options depending on the element selected:

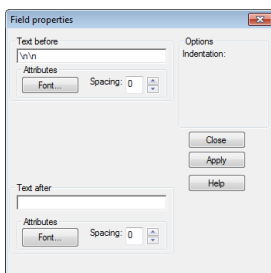
- Formatting header and language units (» [page 48](#))
- Formatting language entries and subentries (» [page 49](#))
- Formatting fields (» [page 50](#))
- Entering and formatting static texts (» [page 51](#))

You can close the window by clicking **Close**.

To open it again, select a field or item of static text in the **Fields in layout** area. In the context menu, select **Properties**.

Formatting header and language units

For each of the four units (header, source language, target language, additional languages), you can specify texts or characters which frame all the fields of the unit.



Field properties for main units

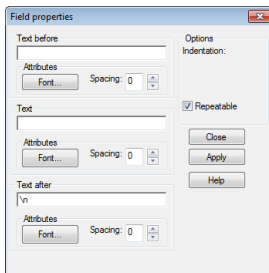


Area	Setting	Meaning
Text before		Text that TermStar displays before the header or the language, e.g. before the source language.
	Font	Text formatting.
Text after	Spacing	Spacing to the previous field.
	Font	Text formatting.
	Spacing	Spacing to the last unit field.

Formatting the main units

Formatting language entries and subentries

You can define text or characters for each language entry and subentry which frame all the fields of the entry.



Field properties for language entries and subentries

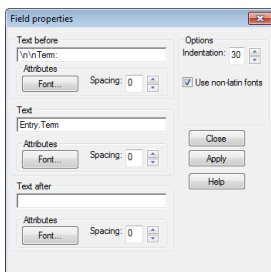
Area	Setting	Meaning
Text before		Text that TermStar displays before the fields of the entry.
	Font	Text formatting.
Text	Spacing	Spacing to the previous field.
	Font	Text formatting.
	Spacing	Spacing between <u>multiple</u> entries in the same data record.
Text after	Font	Text formatting.
	Spacing	Spacing to the last entry field.

Formatting a language entry or subentry

Area	Setting	Meaning
Option	Repeatable	<ul style="list-style-type: none"> <li>Option selected: TermStar displays all the target language entries of a data record. Example: ENG: carrot DEU: Karotte DEU: Möhre</li> <li>Option deselected: TermStar displays one target language entry only. Example: ENG: carrot DEU: Karotte</li> </ul> <p>This option is not relevant for the source language entries as TermStar displays all the source language entries.</p>

Formatting a language entry or subentry (cont.)

**Formatting fields** For each field, you can use the field options to define the font formatting, spacing, indentation, and texts displayed before and after.



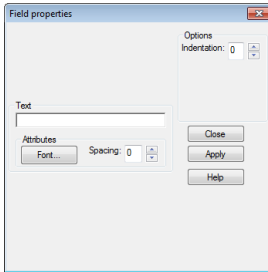
Field properties for fields

Area	Setting	Meaning
Text before		Text that TermStar displays before the contents of the field, e.g. fixed designation such as "part no." or a variable for field names ( <i>» Variables</i> , page 51).
	Font	Text formatting.
	Spacing	Spacing to the previous field.
Text		Layer and name of the field selected, e.g. <Entry.Context>
	Spacing	Spacing to the previous field or (if present) to the text before the field contents.
Text after		Text that TermStar displays after the contents of the field.
	Font	Text formatting.
	Spacing	Spacing to the field contents.
Option	Indentation	Spacing to the left margin.

Formatting a field

Entering and formatting static texts

You can use the options for static texts to determine the contents, font formatting, spacing and indentation for static texts.



Field properties for static texts

Area	Settings	Meaning
Text		Text or characters that TermStar displays as a static text.
	Font	Text formatting.
	Spacing	Spacing to the previous field.
Options	Indentation	Spacing to the left margin.

Formatting a static text

Variables You can also use the following variables for **Text before**, **Text** and **Text after**:

Variable	Meaning	Example
\a	Abbreviated subentry type (» <a href="#">Prefixes for field types</a> , page 105).	<i>Abbr.</i>
\c	Language code	<i>DEU</i>
\f	Field name	<i>Status</i>
\l	Language name (in dialog language)	<i>German</i>
\n	Line break	

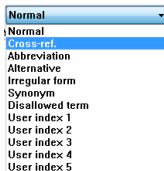
Variables for text fields

Formatting automatic cross-references

TermStar displays subentries in the dictionary as separate entries that refer to the main entry using a cross-reference.

You can define how TermStar displays these subentries and their cross-references.

- Additional units



Units list box

The units **Cross-reference**, **Abbreviation**, **Alternative**, **Irregular form**, **Synonym**, **Disallowed term** and **User index 1 - 5** correspond to the subentries in the normal layout section. Their **Term** field therefore contains synonyms, alternatives, etc. As this field displays the index field for the subentries, this field is mandatory for these sections of the layout.

- **Source language**

As the index field for the main entry, the **Term** field for the source language creates the cross-reference to the main entry. This field must therefore be included if you wish to create an automatic cross-reference.

The remaining fields of the source language correspond to those in the main entry. You can add them if you want TermStar to display further fields of the main entry for the subentry without the user having to use the cross-reference.

- **Target language**

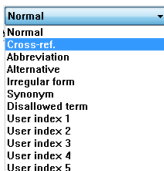
You can use these fields to display all the target language data. The automatic cross-reference, however, can only be used in the source language.

- **Additional languages**

The **Additional languages** layer is not available for subentries.

### How do I edit the cross-reference area of the layout?

1. To open the cross-reference area, select the **Cross-reference** option in the list box in the tool bar:



TermStar displays the available fields and the fields you have selected for the cross-reference area of the layout.

2. Add the **Term** field from the **Cross-reference** layer if this field is not already included in the layout.
3. Add the **Term** field from the **Source language** layer.  
This field generates an automatic cross-reference to the main entry for the subentry.
4. In the **Field properties** window for this field, enter a string as **Text before** for TermStar to display before the cross-reference (e.g. *see or --->*).
5. Format the field so that it can be identified as a cross-reference (e.g. different font colour green).
6. Add additional fields if necessary.

TermStar uses this layout for subentries with an automatic cross-reference to the main entry.

## General layout settings

You can define general settings for each layout. These apply to both areas of a layout ("normal" and "cross-reference").

You have the following options for this:

Area	Meaning
<b>Display graphics</b>	Displays the graphics linked in the <b>Multimedia</b> field: <ul style="list-style-type: none"> <li>● Show original size</li> <li>● Fit to page width</li> <li>● Shrink to page width</li> <li>● Show filename</li> </ul>
<b>Cross-reference colour</b>	Colour in which TermStar displays the <u>manual</u> cross-references. Manual cross-references are cross-references that have been created using the <b>Create cross-reference</b> function or the <b>Create special reference</b> function.  This does not affect how <u>automatic</u> cross-references are displayed (» <a href="#">Formatting automatic cross-references</a> , page 51).
<b>Spacing</b>	You can specify the following distances: <ul style="list-style-type: none"> <li>● <b>Data record distance:</b> Distance between two data records</li> <li>● <b>Left margin:</b> Distance to the left margin of the page</li> <li>● <b>Right margin:</b> Distance to the right margin of the page</li> </ul>
<b>Show data records</b>	How data records are displayed: <ul style="list-style-type: none"> <li>● <b>One data record per page:</b> TermStar displays each data record on a new page.  Especially useful for layouts for the right-hand page. For example, it enables you to display word pairs on the left-hand page and detailed information about the selected data record on the right-hand page.</li> <li>● <b>Show empty fields:</b> TermStar displays the fields of the layout even if they do not contain any information.</li> <li>● <b>Separator:</b> Separator between units, fields and field parts (text before, text, text after)  TermStar displays the separator in the layout preview and when the layout is used in edit mode.</li> </ul>
<b>Date/time format</b>	How the date fields are displayed: <ul style="list-style-type: none"> <li>● <b>Short format:</b> TermStar displays the date in short format (e.g. 29.05.2012 instead of <i>Thursday, 29th May 2012</i>)</li> <li>● <b>Show time:</b> TermStar displays the time as well as the date.</li> </ul>

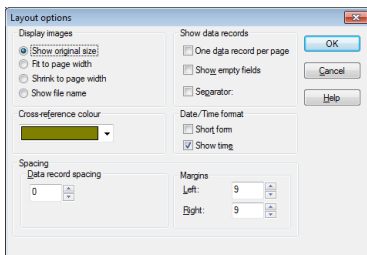
General layout settings

### How do I change the layout settings?

1. In the toolbar of the layout editor, click the **Layout settings** button:



TermStar displays the following window:



2. Change the settings and confirm with **OK**.

Transit closes the window and applies the settings immediately. You can see some of the effects of your settings in the **Preview** area.

## 6 Using start parameters

As an experienced user, you can use start parameters to define which settings Transit uses when it starts up (dialog language, user, user role) and which project/language pair should be opened automatically.

You can use these parameters when opening Transit via a desktop shortcut, in the command line or in a batch file.



### Tip: Specifying the “Action on startup” in “User preferences”

You can also specify the dialog language as well as settings for opening projects and language pairs directly in Transit (Transit's user preferences | **Startup settings**).

Specifying the dialog language:  
-DialogLanguage

This parameter starts Transit with the specified dialog language.

Without this parameter, Transit starts with the dialog language specified in Transit's user preferences (**Startup settings**, option **Dialog language for next startup**).

Syntax	-DialogLanguage=<LanguageCode>
Attribute	<ul style="list-style-type: none"> <li>● &lt;LanguageCode&gt; Dialog language</li> <li>Possible values:             <ul style="list-style-type: none"> <li>• CHS: Chinese</li> <li>• CSY: Czech</li> <li>• DEU: German</li> <li>• ENG: English</li> <li>• ESP: Spanish</li> <li>• FRA: French</li> <li>• ITA: Italian</li> <li>• JPN: Japanese</li> <li>• SVE: Swedish</li> </ul> </li> </ul>
Example	<ul style="list-style-type: none"> <li>● Start Transit with Italian as the dialog language: "c:\Program Files\Transit NXT\bin\transitnxt.exe" -DialogLanguage=ITA</li> </ul>

Specifying the user: -U This parameter starts Transit with the specified user or displays a dialog window in which you can select the user.

Without this parameter, Transit starts with the current Windows user.

Syntax	<code>-U[&lt;UserShortName&gt;]</code>	
Attribute	<ul style="list-style-type: none"> <li>● <code>&lt;UserShortName&gt;</code> Short name of the user (optional)</li> </ul>	
	Note:	The attribute value ignores case and does <u>not</u> distinguish between user names that differ only in terms of case.
	Default:	Transit displays the <b>Select user</b> window and waits for the user to make a selection.
	Note:	The short name is <u>not</u> the “full name” that is displayed as the folder name and in Transit.
Example	<ul style="list-style-type: none"> <li>● When starting Transit, display the dialog window for selecting the user:  <code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" -U</code></li> <li>● Start Transit with user lpb:  <code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" -Ulpb</code></li> </ul>	

Specifying the user role: -H This parameter starts Transit with the specified user role.  
 Without this parameter, Transit displays the **Select user role** window or starts with the role that was selected.

Syntax	<code>-H&lt;role&gt;</code>	
Attributes	<ul style="list-style-type: none"> <li>● <code>&lt;role&gt;</code> User role</li> </ul>	
	Possible values:	<ul style="list-style-type: none"> <li>• 1: Project Manager role</li> <li>• 2: Project Calculation Manager role</li> <li>• 3: Translator role</li> <li>• 4: Reviewer role</li> <li>• 5: Markup Specialist role</li> <li>• 6: Reference Material Manager role</li> <li>• 7: Alignment Specialist role</li> <li>• 8: Terminology Manager role</li> <li>• 9: Terminologist role</li> <li>• 10: Terminology Translator role</li> <li>• 11: Localisation Specialist role</li> <li>• 12: Super User role</li> </ul>
	Note:	TermStar only supports roles 8, 9, 10 and 12.
Examples	<ul style="list-style-type: none"> <li>● Start Transit with the Reviewer role:  <code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" -H4</code></li> <li>● Start Transit with the Super User role:  <code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" -H12</code></li> </ul>	



**Open project: -P** This parameter opens the specified project.  
 Without this parameter, Transit carries out the action specified in Transit's user preferences (**Startup settings**, option **Action on startup**).

Syntax	<code>-P[&lt;ProjectName&gt;]</code>	
Attribute	● <ProjectName>	Path and name of the project file you want Transit to open (optional)
	Note:	The attribute value ignores case and does <u>not</u> distinguish between user names that differ only in terms of case.
	Default:	The last-opened project. If you start Transit with different users (parameter » <a href="#">Specifying the user: -U</a> , page 56), Transit opens the last project that was opened by the current user.
Examples	● Start Transit with project <code>NXT_Word</code> :	<code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" "-Pd:\Transit_NXT\config\global\Nxt_Word.PRJ"</code>
	● Start Transit with the project that was last opened by the current user:	<code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" -P</code>

**Open language pair: -O** This parameter opens the language pairs of the project that is currently open.  
 If you do not specify this parameter, Transit will not open any language pair.

Syntax	<code>-O[&lt;LangPair&gt;]</code>	
Attribute	● <LangPair>	Name of the language pair you want Transit to open (optional)
	Note:	With value <code>*</code> , Transit opens all language pairs in the project (except translation extract).
	Default:	Last-opened language pair for the current project If you start Transit with different users (parameter » <a href="#">Specifying the user: -U</a> , page 56), Transit opens the language pair that was last opened by the current user.
Examples	● Start Transit with project <code>NXT_Word</code> and open its language pair <code>Intro</code> :	<code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" "-Pc:\Users\Public\Documents\Transit NXT\config\global\Nxt_Word.PRJ" -OIntro</code>
	● Start Transit with project <code>NXT_Word</code> and open its last-opened language pair:	<code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" "-Pc:\Users\Public\Documents\Transit NXT\config\global\Nxt_Word.PRJ" -O</code>
	● Start Transit with project <code>NXT_Word</code> and open all its language pairs:	<code>"c:\Program Files\Transit NXT\bin\transitnxt.exe" "-Pc:\Users\Public\Documents\Transit NXT\config\global\Nxt_Word.PRJ" -O*</code>

**Example using all start parameters** The following example starts Transit with the Italian user interface, the user `con` and the user role `Super User`. After the start, Transit opens the project `Brochure_V4-2` and all its language pairs:

```
"c:\Program Files\Transit NXT\bin\transitnxt.exe" -DialogLanguage=ITA -Ucon -H12
-P"c:\Users\Public\Documents\Transit NXT\config\global\Brochure_V4-2.PRJ" -O*
```

# 7 Managing database links

TermStar accesses the databases in which dictionaries are stored by means of *database connections*.

You can use the TermStar user interface to carry out most tasks for managing the database functions (» [TermStar User Guide](#)).

Advanced users can also make use of small programmes for the following special tasks:

- Transferring database connections to other computers (» [page 58](#))
- Finding and deleting incorrect database links (» [page 61](#))

## Transferring database connections to other computers

**What you should know here** If you have set up a TermStar database on the server, you must set up a database connection to the server on each client machine so that all TermStar users can access the database.

To simplify that task, you can transfer the database connection settings from one computer to another. In other words, you set up the database connection on one computer and then transfer the settings to the other computer (target computer).

There are two programs available for this:

- `ODBCDataSaver.exe`: You use this program on the first computer to save the database connection settings to a file.
- `ODBCDataLoader.exe`: You use this program on the target computer to set up the database connection using the saved settings.



### **SQL: Target computers must support Microsoft SQL Server**

The support of SQL databases must be installed on the target computers to establish the connection.



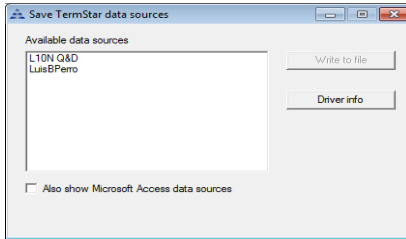
### **Access: Transfer only makes sense with a shared, centralised mdb file!**

Transferring the connection settings for Microsoft Access databases only makes sense if all computers use the same path to access the same central `.mdb` file.

Save ODBC settings to file

**How do I save the settings on the first computer?**

1. Start the program `ODBCDataServer.exe` by double-clicking it.  
The program is located in the `\bin` folder of the Transit installation.  
The program displays the following window:



The program shows all database connections.

By default, the program displays all connections to SQL databases.

In order to display connections to Access databases as well, select **Also show Microsoft Access data sources**.

2. Select the connections that you want to transfer.
3. Click **Write to file**.

TermStar displays the **Save as** window.

- The program suggests the `odbc_data_exchange.ini` file in the `\bin` folder.  
If you want to save the settings in a different folder or file, select the folder or file of your choice.

Click **Save** to confirm your entry.

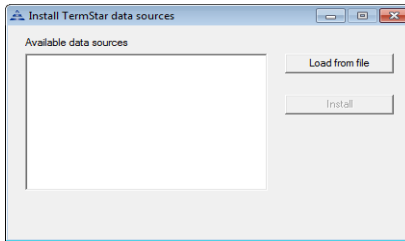
To exit the program, click the **X** on the window title bar.

Set up ODBC settings on the target computer

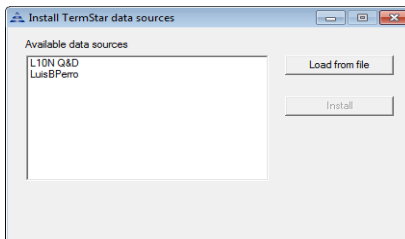
**How do I use the saved settings on the target computer?**

1. Copy the `.ini` file onto the target computer or to a network folder the target computer can access.
2. Start the program `ODBCDataLoader.exe` by double-clicking it.  
The program is located in the `\bin` folder of the Transit installation.

The program displays the following window:



3. Click **Load from file**, select the `ini` file and confirm your selection with **Open**.  
The program shows all saved database connections:



4. Select the connections that you want to set up on the target computer.
  5. Click **Install**.
    - If a database connection of the same name already exists on the target computer, the program displays a message.  
To overwrite the existing database connection, click **Yes** in the message box.
- To exit the program, click the **X** button on the window title bar.

## Finding and deleting incorrect database links

The program *ODBC Data Cleanup* checks database connections and displays all connections with errors.



### Only the ODBC connection is deleted; the database file remains

This step only deletes the ODBC connection to the database.

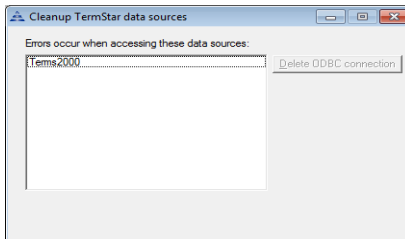
The database file itself is retained, meaning that you can establish a connection again if necessary.

### How do I find and delete incorrect database links?

1. Start the program `ODBCDataCleanup.exe` by double-clicking it.

The program is located in the `\bin` folder of the Transit installation.

The program displays the following window:



The program shows all database connections with errors.

2. Select the connections that you want to delete.
3. Click **Delete ODBC connection**.  
The program displays a precautionary question.
4. To delete the selected connections, click **Yes** in the message box.  
To exit the program, click the **X** button on the window title bar.

# 8 Customising Transit / TermStar

Almost all Transit and TermStar adaptations can be made using the user interface ([» Transit User Guide](#) and [» TermStar User Guide](#)).

This section describes a few exceptions where you will need to edit files in order to customise Transit or TermStar.

- Segmentation by sentence: Correcting abbreviation lists ([» page 63](#))
- Spellchecking: Correcting a list of unknown words ([» page 65](#))
- TermStar: Customising index buttons ([» page 66](#))

## Segmentation by sentence: Correcting abbreviation lists

What you should know here

In the case of segmentation by sentence, a segment marker is set after a dot marking the end of a sentence. However, a dot can also be placed after an abbreviation that is not to be segmented by. Abbreviation lists are used to distinguish whether a string is an abbreviation (without subsequent segmentation) or a “normal” word (with subsequent segmentation).

During project import, the abbreviation lists can be expanded by using the interactive abbreviation check (**Segmentation** project setting, **Check abbreviations for segmentation during import** option, » [Transit User Guide](#)).

The abbreviation lists are Unicode-encoded text files. Therefore you can open and edit the files with a Unicode text editor, e.g. to delete or correct incorrect entries.



### TEXT EDITOR MUST SUPPORT UTF-16!

**Only edit the files using a text editor that supports UTF-16.**

Otherwise, the abbreviation lists may be saved with incorrect coding and may lead to incorrect abbreviation check results or to unwanted segmentation.

Filename of abbreviation lists

Transit saves the abbreviation lists for each source language in two files:

- `<language code>_neg.ewl`

This list contains strings that are abbreviations. Transit does not segment at this point.

Filename example for English (UK): `eng_neg.ewl`

- `<language code>_pos.ewl`

This list contains strings that are normal words (i.e. no abbreviations). This prevents the user from having to check these strings again in future abbreviation checks.

Filename example for English (UK): `eng_pos.ewl`

Scopes and storage locations

During the interactive segment check, the user can define the scope that applies for the lists. The lists are stored in different folders according to their scope:

- **Global** scope (for all projects and customers): `config\global` folder
- **Customer** scope (for all projects for the current customer): `config\customers\<customer>` folder
- **Project** scope (for the current project only): Working folder of the project

If available, Transit uses the lists from all three scopes for a project, e.g. global, customer-specific and project-specific lists.

- Interaction of ewl files and scopes
- During import, Transit creates a list of potential abbreviations and checks for each of these strings whether it is contained in one of the \*\_neg.ewl files.
- If this is the case, Transit treats the string as an abbreviation and does not segment at this point.
- Otherwise, Transit checks for each string whether it is contained in one of the \*\_pos.ewl files.
- If no, Transit displays the string in the abbreviation check. The user can decide interactively whether it is an abbreviation.
  - If yes, the string is not displayed in the abbreviation check.



**Tip: Join segments virtually**

Despite careful abbreviation checking, it may happen that a sentence is segmented in an undesirable point. This is usually caused by a normal word that is exceptionally used as abbreviation (e.g. “*Dominic*” as abbreviation in “*Dominic. Republic*”).

In this case, you can virtually join target language segments in the Transit editor. The segments then form a unit and can be translated “in one piece” (» [Transit User Guide](#)).



## Spellchecking: Correcting a list of unknown words

**What you should know here** When performing a spellcheck in the Transit editor, you can add unknown words to a list so that they will not be displayed as errors in the future (» [Transit User Guide](#)).

The list of unknown words is an alphabetically sorted, Unicode-encoded text file. Therefore you can open and edit the file with a Unicode text editor, e.g. to remove words that have been added accidentally.



### TEXT EDITOR MUST SUPPORT UTF-16!

**Only edit the file using a text editor that supports UTF-16.**

Otherwise, the list of unknown words may be saved with incorrect coding and may lead to incorrect results during the spellcheck.

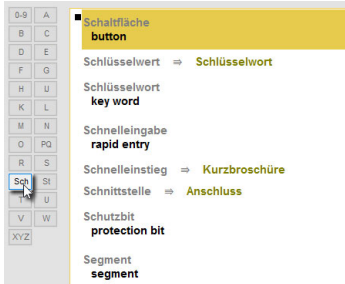
**Filename and storage locations** Transit saves the list of unknown words for each language in a file. The name of the file and where it is saved depend on the basis of the spellcheck:

- Based on MS Word or reference files
  - These lists of unknown words are project-specific and apply for the current project only.
  - Folder: Working folder of the project
  - Filename: <project name>\_<language code>.rsa  
Filename example for Italian and "NXT\_Word" project: `nxt_word_ita.rsa`
- Based on OpenOffice (Hunspell) or project dictionaries
  - These lists of unknown words are user-specific and apply for all of the current user's projects.
  - Folder: User folder (`config\users\<user name>`)
  - Filename: `SpellUserDic.<language code>`  
Filename example for Italian: `SpellUserDic.ita`

## TermStar: Customising index buttons

What you should know here TermStar can display index buttons to navigate within the dictionary (» [TermStar User Guide](#)).

You can individually configure the index buttons for each source language (e.g. additional button for numerals or **Sch** and **St** buttons for German):



With the additional **Sch** index button you can navigate to the first term beginning with "Sch".

Configuration files The index buttons for each language are configured in a separate file.

- Folder: `config\global`
- Filename: `buttons.<language code>`

Filename example for German: `buttons.deu`

In the `buttons.def` file, the default index buttons are configured. These are used for source languages that do not have their own configuration file.

Content of configuration files You can create and edit the configuration files with a text editor. They are structured as follows:

- First line: Fixed content [`Register`]
- Subsequent lines: Button definitions with the following syntax:  
`<n>=<Label> <CharGroup>*`
  - `<n>`: Consecutive number for the position of the index button
  - `<Label>`: Caption for the index button
  - `<CharGroup>`: Initial character or character string of the terms to which the index button should navigate (case-insensitive).

Example:

[Register]	
1=0-9 0*	The 1 <sup>st</sup> button displays <b>0-9</b> and navigates to the first term beginning with a zero.
2=A A*	The 2 <sup>nd</sup> button displays <b>A</b> and navigates to the first term beginning with A.
...	
19=Sch SCH*	The 19 <sup>th</sup> button displays <b>Sch</b> and navigates to the first term beginning with <i>Sch</i> .
20=St ST*	The 20 <sup>th</sup> button displays <b>St</b> and navigates to the first term beginning with <i>St</i> .
...	
25=XYZ X*	The 25 <sup>th</sup> button displays <b>XYZ</b> and navigates to the first term beginning with X.

Extract from a customised configuration file for German index buttons  
 The navigation does not consider the case.



**Changes are not displayed until the source language has been changed**

TermStar displays new index buttons or ones that have been changed when you reselect the source language after making the change.

The quickest way of doing this is by swapping the source and target language twice (with key combination CTRL + A).

# 9 Organising reference material

## Copying, moving and deleting reference material

What you should know here

The Transit reference material is organised as language pairs. You can use Windows functions to copy, move or delete the language files as you would with any other file.

The **Organise reference material** function supports you with additional options: You can filter and select the reference material based on ten different criteria (e.g. date, file type or Transit version) in order to move it to an archive folder, to copy it to another location or to delete it (» [Three steps of the function](#), page 68).

To ensure that copied or moved files can be used as reference material, the function guarantees that all of the associated language files are copied or moved (» [“All associated language files...” message](#), page 69).



### Tip: “Copy current project to reference folder” function

To save language pairs from a completed project so that they are structured as reference material, you can simply use the **Reference material | Copy current project to reference folder** function.

When doing this, you can also specify how you want the language pairs to be copied to the reference folder and easily eliminate any naming conflicts.

Three steps of the function

The **Organise reference material** function works in three steps:

1. You select the folder that contains the reference material (including sub-folders).

Transit scans the reference material for ten criteria.

To ensure that you can access the scan results at a later point, you can save them and load them again (» [Saving and loading the scan results](#), page 74).

2. You define the criteria that Transit uses to filter the reference material.

In doing so, you limit the selection of reference files that are displayed.

3. You select the files that you want to copy, move or delete.

You can find out how to do this in the section » [Using the “Organise reference material” function](#), page 70.

“All associated language files...” message

The reference material from Transit consists of *language pairs* with at least two files (source- and target-language file). With multilingual projects, this also consists of several target-language files.

- Example: The `introduction.docx` file was translated from German into English and Spanish. This resulted in the following reference files: `introduction.deu`, `introduction.eng` and `introduction.esp`.

To ensure that copied or moved files can be used as reference material, it makes sense that all of the associated language files are copied or moved.

- Example: If you want to move the above-mentioned language file `introduction.deu` to another folder, you must also move the `introduction.eng` and `introduction.esp` files so that you can use these as reference material.

Transit therefore displays a message asking whether you want to copy, move or delete all of the associated language files.

- Example: If you want to move the above-mentioned language file `introduction.deu` to another folder, you have the following options:
  - **Yes:** In addition to the `introduction.deu` file, Transit also moves the `introduction.eng` and `introduction.esp` files.
  - **No:** Transit only moves the `introduction.deu` file. The `introduction.eng` and `introduction.esp` files remain unchanged in the previous folder.
  - **Cancel:** Transit cancels the move. All three files remain unchanged in the previous folder.

Only in exceptional cases is it useful to only copy, move or delete the language file for one language:

- Deleting just one language
 

You only want to delete files for one language because you no longer require these as reference material, or they no longer meet your quality standards.

Example: You want to prevent the Spanish translation from being used for future pretranslations. To do this, delete only the `introduction.esp` file. You can then continue to use the other two files for German-English translation projects.
- Copying only certain language combinations
 

You want to create a copy of the reference material, but not for all languages.

Example: You want to create a copy from the multilingual reference material that only contains German-Spanish language pairs. To do this, copy only the `introduction.deu` and `introduction.esp` files.

Using the  
"Organise reference  
material" function



### Creating a new destination folder in advance

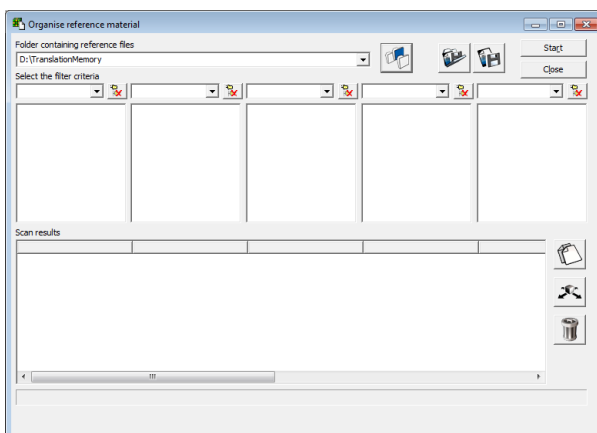
You can use this function to copy or move reference material to an existing folder.

If you want to use a new folder as the destination folder, create this folder before you call up this function.

### How do I organise my reference material?

1. Select **Reference material | Organise reference material** from the resource bar.

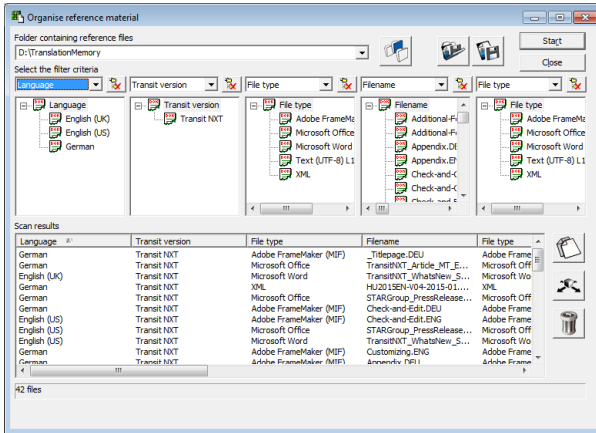
Transit displays the following window:



2. Select the reference material you want to organise:
  - Click the **Select folder containing reference material** symbol.
  - Select the folder and confirm your selection with **OK**.
3. Click **Start**.



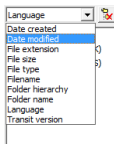
Transit scans all of the reference files in the specified folder and its sub-folders:



Middle: Five selected filter criteria and their values in the scanned reference files

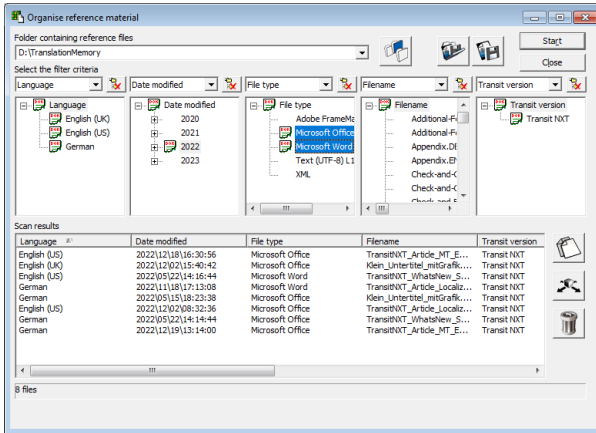
Bottom: Scan result as a list and the number of files that correspond to the selected filter criteria

4. If you want to filter for other criteria, select this:



You can filter for ten criteria, e.g. for the date of the last change.

- If you want to limit the selection, select the values for the individual filter criteria:



Transit updates the scan result in accordance with your filter criteria.  
 Example: Transit only displays files that have the date of last change as 2022 and with file types Microsoft Office or Microsoft Word.

To select several values, press and hold the CTRL button while clicking the values.  
 Normally, all values from the subfolders are included.

Example: If you select the year 2022, the selection automatically applies for all subentries from 01 to 12 (i.e. for all months).

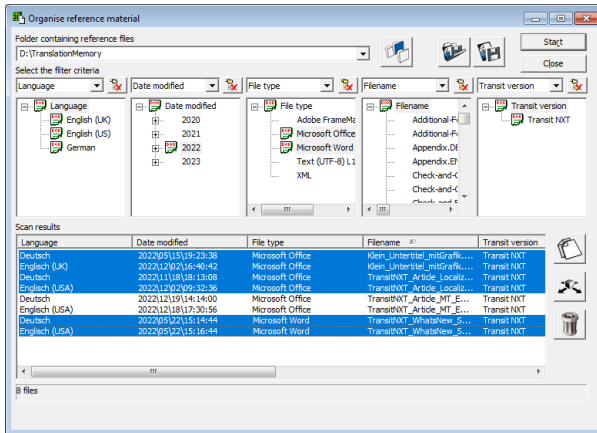
- You can use the plus and minus signs to display and hide the subfolders.
- For the Folder hierarchy criterion, it may make sense to only take into consideration the files from the selected level. To do this, click the **Exclude subfolders** symbol.



To sort the scan results, click in the header of the column according to which you want to sort the values.



6. In the **Scan results** section, select the files that you want to delete, copy or move.



To select several files, press and hold the CTRL button while clicking the corresponding rows.

If you want to delete, copy or move all of the files that are displayed, you do not have to select anything.

7. Decide what you want to do:



- Copy selected files: Click the **Copy** symbol. Select the destination folder and confirm your selection with **OK**.

Transit copies the files into the specified folder.



- Move selected files: Click the **Move** symbol. Select the destination folder and confirm your selection with **OK**.

Transit moves the files to the specified folder and deletes them from the original folder.



- Delete selected files: Click the **Delete** symbol.

Transit deletes the files from the specified folder.

8. Transit may display a message asking whether you want to copy, move or delete all of the associated language files (» **“All associated language files...” message**, page 69).

You have the following options:

- **Yes:** Transit copies, moves or deletes all of the associated language files.
- **No:** Transit copies, moves or deletes only the selected language file; all others remain unchanged.
- **Cancel:** Transit cancels the process of copying, moving or deleting and leaves all language files unchanged.

If you have finished your tasks in the **Organise reference material** window, click **Close**.

**Saving and loading the scan results** The process of scanning the reference material may take some time, depending on the volume. To ensure that you can access the results at a later point, you can save them and load them again.



### How do I save the scan results?

1. Click the **Save current results** symbol in the **Organise reference material** window.
2. Specify a name for the file and the folder to which you want to save it. Confirm your entry by clicking **Save**.

Transit saves the scan results in a file with the file extension dat.



### How do I load scan results that have been saved?

1. Click the **Open results file** symbol in the **Organise reference material** window.
2. Select the dat file with the saved analysis results. Confirm your selection with **Open**.

Transit displays the loaded scan results.

## Modifying reference material

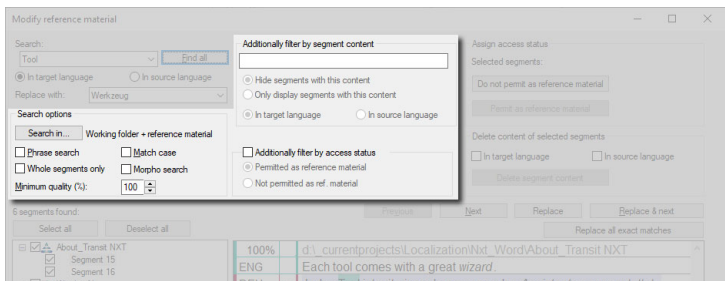
The Transit reference material is organised as language pairs. You can therefore open and edit the reference material – as is the case with each language pair – in the Transit editor (» [Transit User Guide](#)).

The **Modify reference material** function goes even further: Using search options and filter criteria, you can replace selected content, empty segments, and exclude segments from being used as reference material.

In addition, you can save the list of found segments for subsequent evaluation (» [Saving result lists](#), page 80).

### Search options and filter criteria

You can specify the following settings in the **Modify reference material** window:



Search options and filter criteria for the precise maintenance of reference material

- **Search in...:** Specify the language pairs in which you want Transit to search:
  - **Reference material:** Transit searches in the reference material for the current project.
  - **Working folder:** Transit searches in the language pairs in the working folder for the current project.
  - **Both:** Transit searches in the reference material and in the working folder for the current project.

In addition, you can specify the minimum segment status that you want Transit to take into consideration during the search.

- **Phrase search:** With this option Transit searches for text that exactly matches the search text.
- **Match case:** With this option Transit takes account of upper / lower case.
- **Whole segments only:** With this option you specify that the search text must match the entire segment content.
- **Morpho search:** With this option Transit also finds inflected forms of the search text.

- **Minimum quality:** With this option you can define how similar matches should be (not relevant for phrase search and morphological search).
- **Additionally filter by segment content**  
 Here, you can use segment content to limit which segments are taken into consideration. This may involve a string or a regular expression.
  - **Hide segments with this content:** Transit only displays segments without this content.
  - **Only display segments with this content:** Transit only displays segments with this content.
 In addition, specify whether the filter criterion relates to the source language or the target language.
- **Additionally filter by access status:** Here, you can specify whether Transit should only display segments that are permitted or not permitted as reference material.

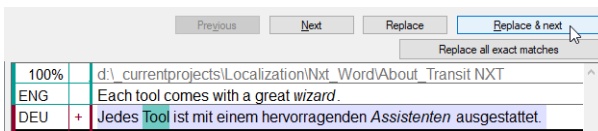
Exact matches and fuzzy/morpho matches

Depending on the search options, Transit not only finds exact matches, but also segments with similar content:

- Matches with a different word order
- Fuzzy matches with a quality below 100%
- Matches from the morphological search

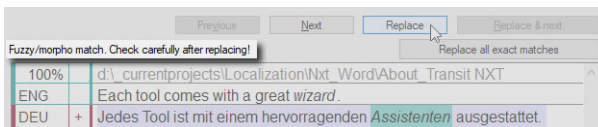
For these matches, you usually have to check and adjust the segment after replacing it. Transit therefore distinguishes between exact matches and other matches:

- For exact matches, you can replace and navigate to the next match with one mouse click:



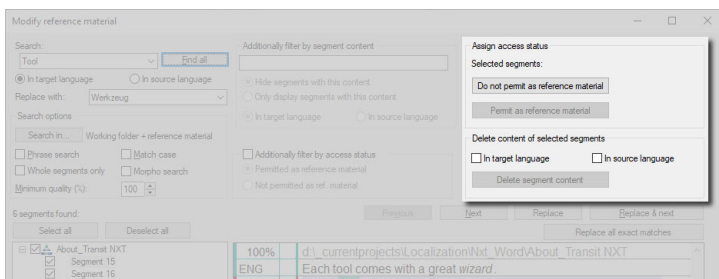
Exact match for "Tool": The **Replace & next** button is also active.

- You can replace all exact matches with one mouse click.
- For non-exact matches, Transit displays a hint text to draw your attention to the necessary check:



Morphological match for "Assistant": Hint reminding to check the segment after replacing.

“Do not permit as reference material” or delete segment content? When modifying the reference material, there are two options for excluding segments from being used as reference material:



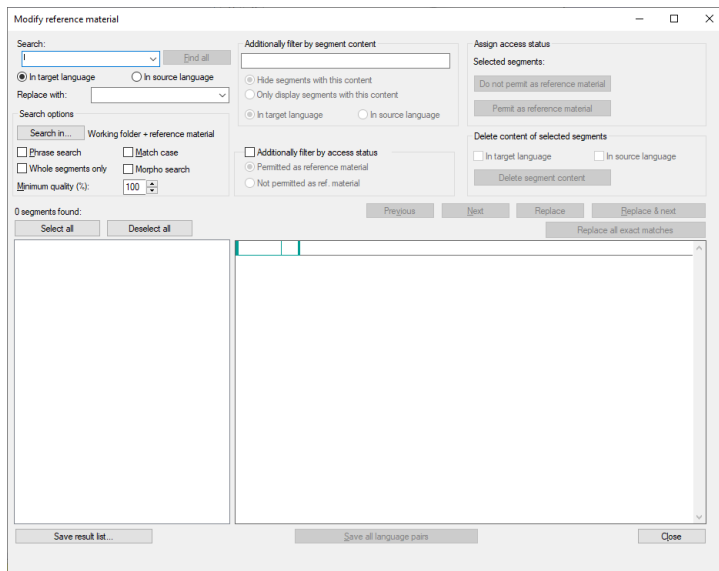
In order not to use a reference segment, you can unpermit it or delete its content.

- Assigning the Do not permit as reference material access status  
 With this function, the segment is no longer used for pretranslation or fuzzy matches but is retained. Transit can continue to display the contents of the segment, e.g. in order to research the context of neighbouring reference segments.  
 Furthermore, you can change the access status back to Permit as reference material if the segment is to be used again later.  
 This allows you to temporarily exclude reference material from use, e.g. during a revision that taking time or until important corrections have been made.
- Deleting the segment content  
 With this function, Transit empties the source and/or target language segment. This means that it can no longer be used for pretranslation or fuzzy matches.  
 This permanently deletes the segment content. Therefore the exclusion as reference material is irreversible.

Using the “Modify reference material” function

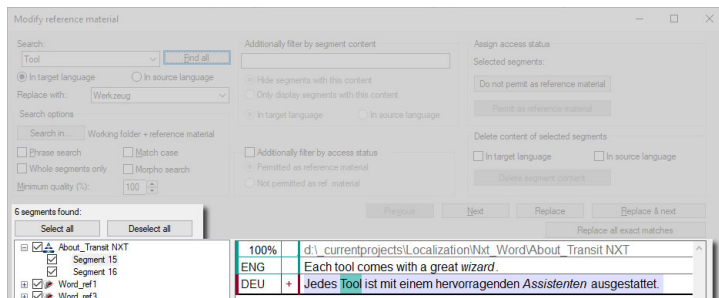
### How do I modify my reference material?

1. Select **Reference material | Modify reference material** from the resource bar.  
Transit displays the following window:






2. Enter the search string in the **Search** field.  
Select you want to search for the string **In target language** or **In source language**.
3. Select the required search options and filter criteria (» [Search options and filter criteria](#), page 75).
4. Click **Find all**.

Transit lists all segments that match the search options and filter criteria:



Bottom left: Number of matches and result list with file and segment numbers of the segments matching with the search and filter options

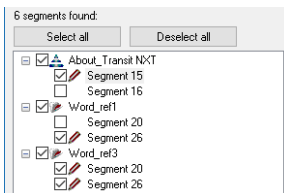
The symbols indicate the origin of the segments:

-  symbol: Language pair of the current project
-  symbol: Reference file of the current project
-  symbol: TM Container of the current project

On the right, Transit displays the respective segment with its origin (path to the reference language pair or TM Container attributes) as well as the source and target language content.

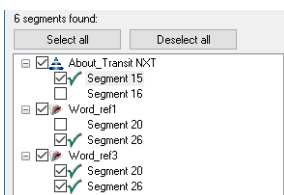
5. You can select or deselect segments:
  - All found segments: Click **Select all** or **Deselect all**.  
After the search, all found segments are selected automatically.
  - All segments of a language or reference file: Click the checkbox on the left of the filename.
  - Single segment: Click the checkbox on the left of the segment number.
6. To navigate through the selected segments, click **Next** or **Back**.  
The search text found is highlighted in green in the segment.
7. Modify the selected segments:
  - Edit segment content manually: Click the segment and edit it.
  - Replace search text in selected segments:
    - In the **Replace with** field, enter the string that you want Transit to replace with.
    - Click **Replace** to replace the search text in the displayed segment.
    - Click **Replace & next** to replace the search text in the displayed segment and navigate to the next selected segment.  
This function is only supported for exact matches (» **Exact matches and fuzzy/morpho matches**, page 76).
    - Click **Replace all exact matches** to replace the search text in all selected segments with exact matches.  
This does not replace matches with a different word order, fuzzy matches below 100% and matches from the morphological search.
  - Change the access status of the selected segments: Click **Do not permit as reference material** or **Permit as reference material** (» **Do not permit as reference material" or delete segment content?**, page 77).
  - Delete segment content of selected segments:
    - Under **Delete content of selected segments**, select **In target language** and/or **In source language**.
    - Click **Delete segment content**.

As soon as you have modified a segment, Transit marks it with a red pen:



Segments with red pen have been changed, but not yet saved.

8. To save your changes, click **Save all language pairs**.



The green checkmark indicates that the changes have been saved.

When you are finished, close the window with **Close**.

**Saving result lists** You can save the list of found segments as a CSV file.

1. Click **Save result list** in the **Modify reference material** window.
2. Specify the folder and name of the CSV file and confirm your selection by clicking **Save**.

Transit saves the result list in the specified file.



## Compacting reference material

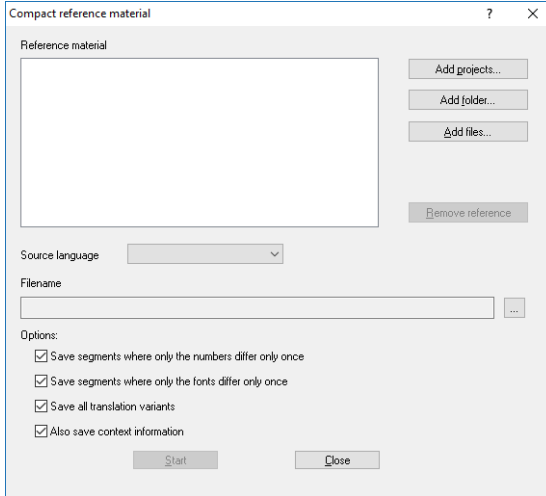
You can use the language pairs that you create while translating in Transit as reference material for future projects. This reference material may contain a large number of identical segments ("*internal repetitions*").

To reduce the data volume, you can compact (compress) your reference material. In doing so, Transit removes any identical segments that appear multiple times or any segments that deviate slightly.

- |                             |  |
|-----------------------------|--|
| Size of the compacted files | If you compact reference material with different file types, Transit generates a separate file for each individual file type. Transit also divides files if they would contain more than 15,000 segments.  |
| Compacting options          | <p>You can use the following options to specify whether Transit should save segments that deviate slightly only once or if it should distinguish between each variant.</p> <ul style="list-style-type: none"> <li>● <b>Save segments where only the numbers differ only once</b> <p>This setting can be useful because Transit can automatically adjust numerical values during pretranslation.</p> <p>It is therefore not usually necessary to save several segment pairs in compacted reference material if these only differ as a result of containing different numbers.</p> </li> <li>● <b>Save segments where only the fonts differ only once</b> <p>This setting can be useful because Transit can automatically adjust fonts during pretranslation.</p> <p>It is therefore not usually necessary to save several segment pairs in compacted reference material if these only differ as a result of containing different fonts.</p> </li> <li>● <b>Save all translation variants</b> <p>This setting is relevant if segments that appear multiple times in the source text have to be translated differently. If you want to include these translation variants in the compacted reference material, select this option.</p> <p>This is useful if you usually use the <b>No pretranslation if variants exist</b> project setting for pretranslation. In this case, Transit displays all of the variants as translation suggestions for the translation.</p> </li> <li>● <b>Also save context information</b> <p>This setting is relevant if, in addition to the segment contents, you also want to take structure information into consideration, e.g. in order to translate headings only with headings and translate the contents of table cells only with table cells.</p> <p>If you want to receive this structure information in the compacted reference material, select this option. This is useful, for example, if you are using context-based pretranslation.</p> </li> </ul> |

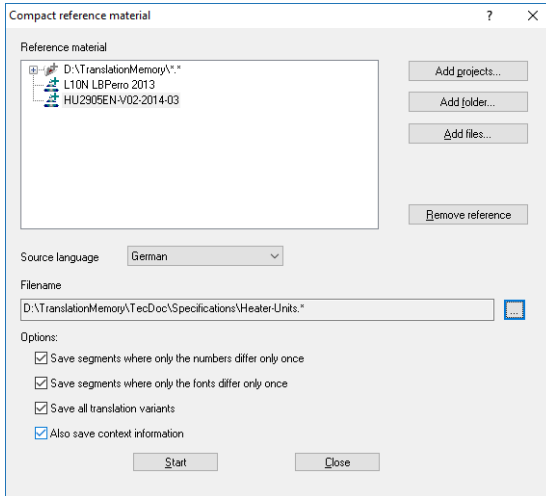
Compacting the reference material **How do I compact reference material?**

1. Select **Reference material | Compact reference material** from the resource bar. Transit displays the following window:



2. Specify the reference material that you want to compact.
  - Language files and reference material from a Transit project: Click **Add projects**, select the project, and confirm your selection with **OK**.
  - Language files in a folder: Select **Add folder**, select the folder, and confirm your selection by clicking **Open**.
  - Individual language files: Select **Add files**, select the files, and confirm your selection by clicking **Open**.
3. Specify the source language of the reference material: To do this, select the appropriate language from the **Source language** list. Transit requires this information to decide whether there are translation variants or not.
4. Specify where you want Transit to save the compacted reference material: To do this, click ... to the right of the **Filename** field. Select the drive and folder, and enter a filename. Confirm your selection by clicking **Save**.

- Specify the compacting options (» [Compacting options](#), page 81).



- Click **Start**.

When Transit has compacted the reference material, it displays the following message:

Completed successfully.

Confirm the message with **OK**.

- Close the **Compact reference material** window by clicking **Close**.

Transit has compacted the reference material. You can now use it as reference material for your projects.

# 10 Open source spellcheck dictionaries

What you should know here As the basis for spellchecking in Transit, you can use open source dictionaries (among others; **Review | Spellcheck | Based on | OpenOffice (Hunspell)**).

When installing Transit, many open source dictionaries are installed for spellchecking. These are saved in the `spell` folder of your Transit installation as files with the extensions `aff` and `dic`.



## OpenOffice.org and spellcheck dictionaries

The *OpenOffice.org project* is an open source project that aims to develop an open-access office suite (» <https://www.openoffice.org/>).

The project also provides dictionaries that Transit can use for spellcheck.

You can add `aff` and `dic` files for additional languages or replace already installed files with new ones. To do this, you can download dictionary packages from the OpenOffice website that contain the required `aff` and `dic` files (» <https://extensions.services.openoffice.org/en/dictionaries>).

Filename of `aff` and `dic` files The names of `aff` and `dic` files are composed of ISO language and country code (» [Supported working languages](#), page 142).

Exception: A few languages do not use a country code.

Language	aff file	dic file	Notes
English (USA)	en_US.aff	en_US.dic	
German (Germany)	de_DE.aff	de_DE.dic	
German (Switzerland)	de_CH.aff	de_CH.dic	
German (Austria)	de_AT.aff	de_AT.dic	
Persian	fa_IR.aff	fa_IR.dic	
Afrikaans	af_.aff	af_.dic	No country code
Catalan	ca_.aff	ca_.dic	No country code

`aff` and `dic` file examples

**Installing an open source spellcheck dictionary** **How do I install an open source spellcheck dictionary?**

1. Make sure that the `aff` file and the `dic` file have the correct filename (» [Filenames of aff and dic files](#), page 84).  
Usually the `aff` and `dic` files in the downloaded dictionary packages are already named accordingly. Otherwise you have to rename the `aff` and `dic` files accordingly. Use the names of the files already installed in the `spell` folder as a guide.
2. Close Transit.
3. Copy the `aff` and `dic` files to the `spell` folder in your Transit installation.
4. Start Transit.

Now the newly installed spellchecker is available in Transit.

**Uninstalling an open source spellcheck dictionary** **How do I uninstall an open source spellcheck dictionary?**

1. Close Transit.
2. Remove the `aff` and `dic` files from the `spell` folder of your Transit installation.  
Make sure that the `aff` file and the `dic` file correspond to the spellcheck language to be removed (» [Filenames of aff and dic files](#), page 84).
3. Start Transit.

Now the removed spellchecker is no longer supported by Transit.

# 11 Transferring TermStar databases from Access to Microsoft SQL Server

What you should know here If you install Microsoft SQL Server on your computer, you can transfer your TermStar databases from Microsoft Access to the SQL server.

In most cases, the TermStar *NXT* Received database shall be transferred. Therefore the following example describes the transfer of this database.

For the transfer, you perform the following steps:

- Renaming the existing ODBC connection ([» page 87](#))
- Creating the new SQL database ([» page 90](#))
- Transferring the database to the new SQL server ([» page 95](#))
- Deleting the connection to the Access database ([» page 97](#))

## Renaming the existing ODBC connection

First rename the existing ODBC connection. You need to do this so that you can use the name of the existing ODBC connection for the new SQL database.



**Use the 32-bit version of the ODBC Data Source Administrator.**

**Ensure that you use the 32-bit version of the ODBC Data Source Administrator**

You cannot use the 64-bit version to rename the ODBC data source for TermStar.

If you are still running Transit under Windows 7 or earlier, see » [Windows 7: Starting the 32-bit ODBC Data Source Administrator](#), page 98.



**Renaming requires administrator rights!**

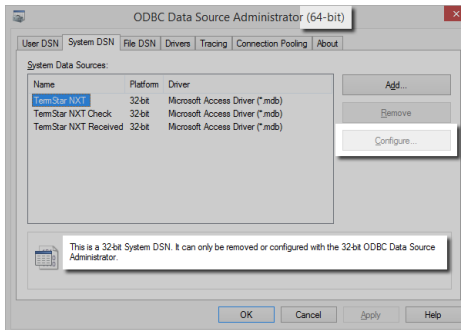
In order to rename the ODBC connection, you must have administrator rights or the corresponding authorisations.

### How do I rename an existing ODBC connection?

1. Start the 32-bit version of the ODBC Data Source Administrator:

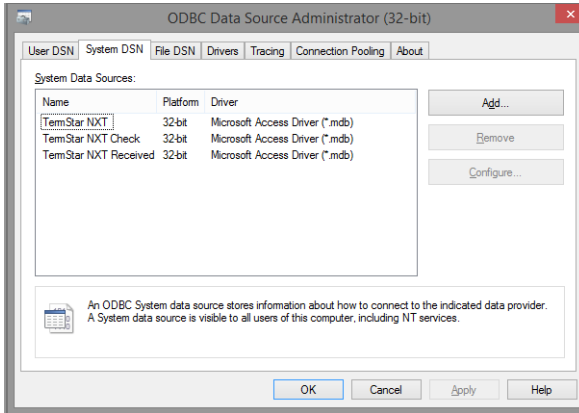
In the Windows Control Panel, select **System and Security | Administrative Tools | ODBC Data Sources (32-bit)**.

If you have accidentally started the 64-bit version of the ODBC Data Source Administrator, the Access databases are not displayed by TermStar or the **Configure** button is not active:



With the 64-bit version, you cannot configure any Access databases from TermStar.

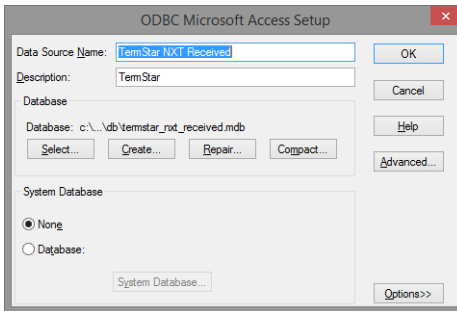
- Switch to the **System DSN** tab and select the datasource **TermStar NXT Received**.



The **System DSN** tab contains the **TermStar NXT**, **TermStar Check** and **TermStar NXT Received** databases, which were created automatically during the installation of Transit.

- If you want to rename a database you have created yourself, you will find it on the **User DSN** tab or on the **System DSN** tab.
- Click **Configure**.

Windows displays the following window:

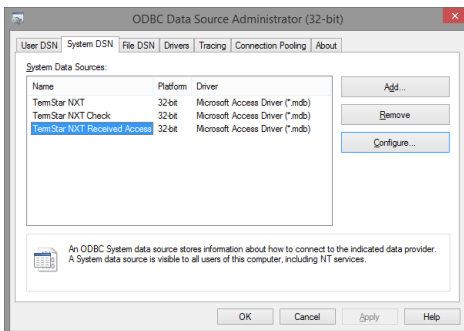


Here you specify the new name.

- In the **Data Source Name** field, change the name of the ODBC connection, e.g. to **TermStar NXT Received Access**, and confirm with **OK**.



Windows displays the changed name:



You have changed the ODBC connection name.

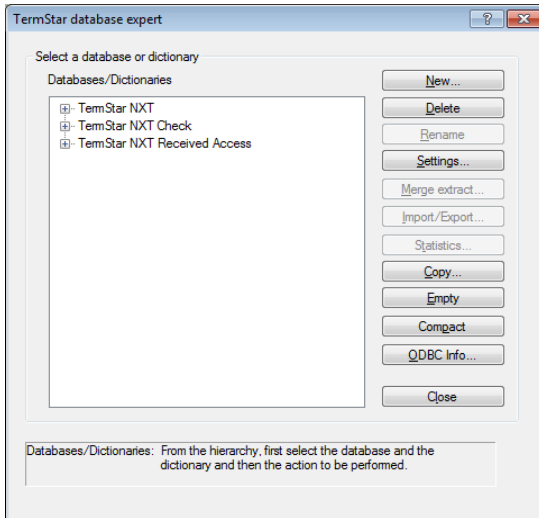
You can now close all the windows of the ODBC data source administrator and the control panel.

## Creating the new SQL database

Next use Transit to create a new SQL database:

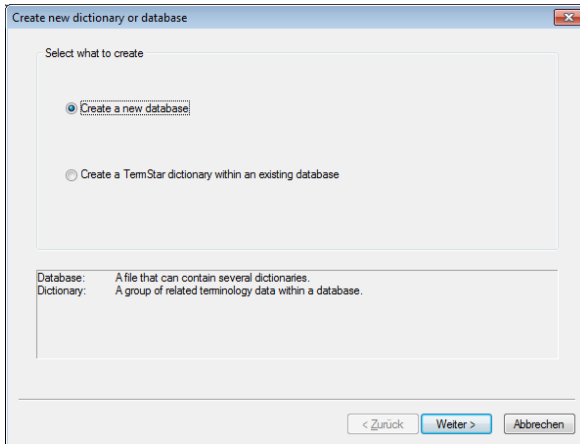
1. In the Transit resource bar, select **Dictionaries | Dictionaries / Databases | Manage dictionaries / databases**.

Transit displays the following window:



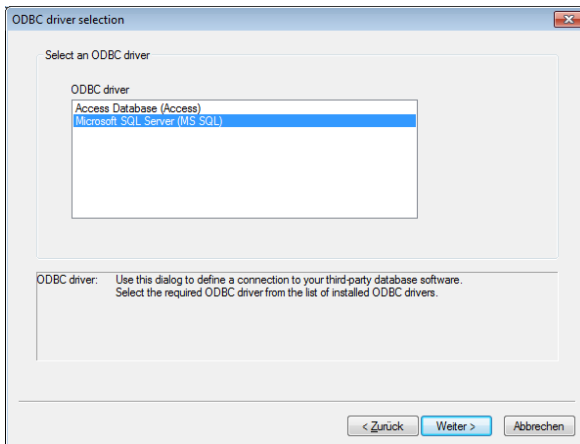
2. Click **New**.

Transit displays the following window:



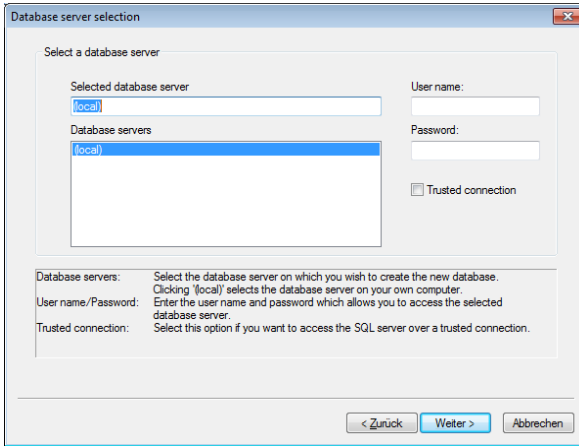
3. Select the **Create a new database** and click **Next**.

Transit displays the following window:



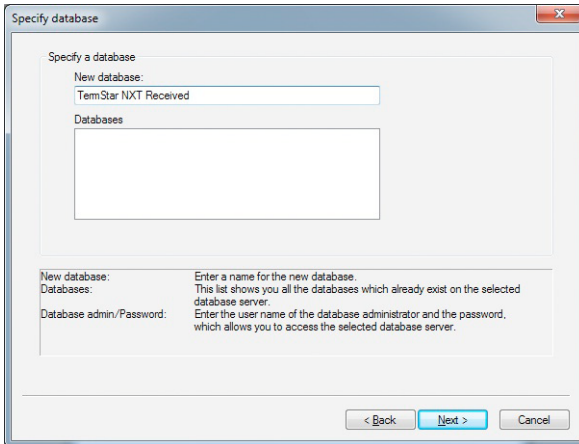
4. Select the **Microsoft SQL Server (MS SQL)** and click **Next**.

Transit displays the following window:



As database server, (local) is automatically selected.

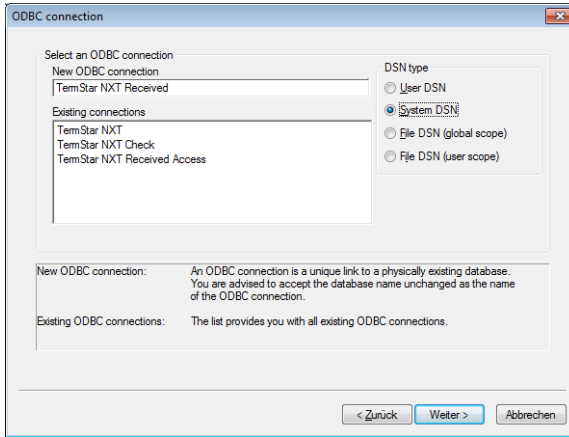
5. Make sure that (local) is selected as database server and click **Next**.  
Transit displays the following window:



Make sure that you enter the name TermStar NXT Received correctly.

6. As **New database** enter TermStar NXT Received and click **Next**.

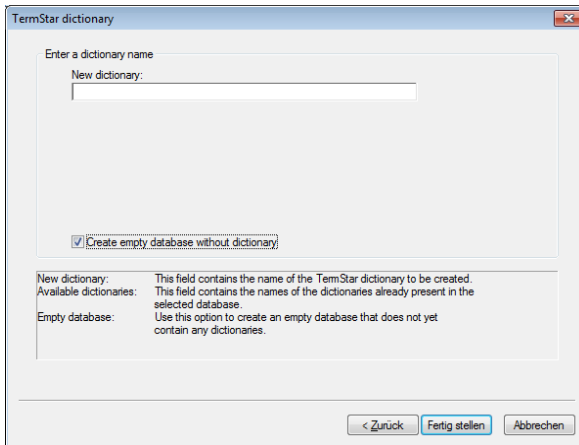
Transit displays the following window:



The ODBC connection must be named `TermStar NXT Received`.

7. Make sure that the name of the ODBC connection is `TermStar NXT Received`.  
Select **System DSN** and click **Next**.

Transit displays the following window:



Create the database without dictionary.

8. Select **Create empty database without dictionary** and click **Finish**.

Transit displays the following message:

Empty database created successfully.

9. Click **OK**.

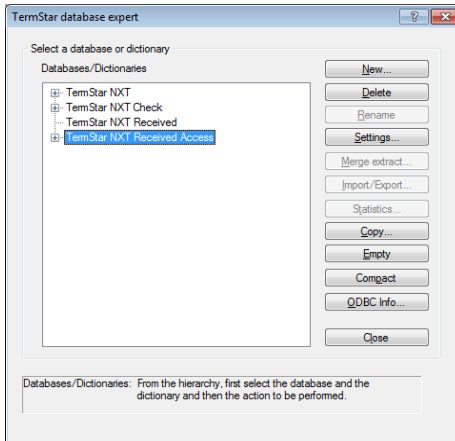
The **TermStar database expert** window now displays the new TermStar NXT Received database.

## Transferring the database to the new SQL server

Then use Transit to transfer the contents of the existing Access database into the newly created SQL database.

1. In the Transit resource bar, select **Dictionaries | Dictionaries / Databases | Manage dictionaries / databases**.

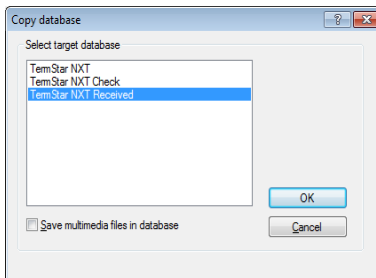
Transit displays the following window:



Here you select the old database as source.

2. Select the old `TermStar NXT Received Access` and click **Copy**.

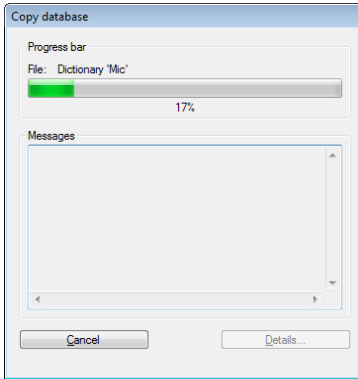
Transit displays the following window:



Here you select the new database as target.

3. Select the new SQL database `TermStar NXT Received` and confirm with **OK**.

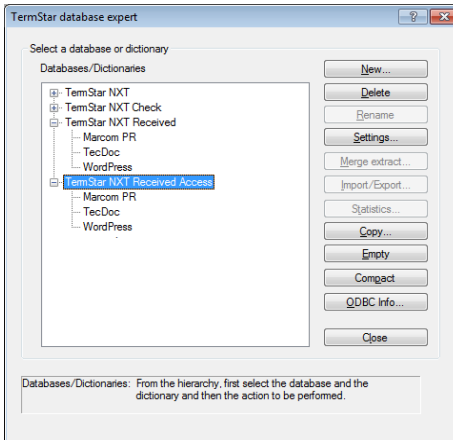
Transit transfers the content of the old TermStar NXT Access database to the SQL database:



Once the transfer is completed, Transit displays the following message:  
Completed successfully.

4. Click **OK**.

The **TermStar database expert** window now displays dictionaries of the old also in the new TermStar NXT Received database:



New TermStar NXT Received database with transferred dictionaries



## Deleting the connection to the Access database

Lastly use Transit to delete the ODBC connection to the Access database. Doing this will prevent you from accidentally working with the old database.



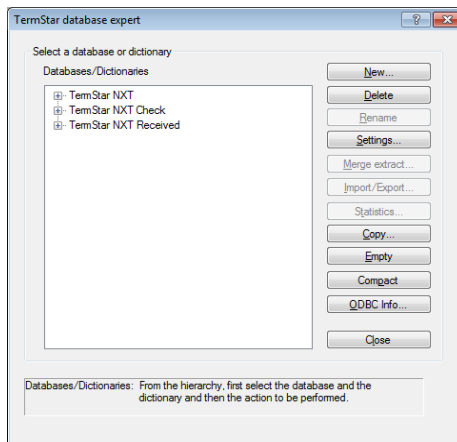
### Only the ODBC connection is deleted; the database file remains

This step only deletes the ODBC connection to the database.

The database file itself (.mdb file) is retained, meaning that you can establish a connection again in case of emergency.

1. In the Transit resource bar, select **Dictionaries | Dictionaries / Databases | Manage dictionaries / databases**.
2. Select the old **TermStar NXT Received Access** database and click **Delete**.  
Transit displays a confirmation prompt.
3. Confirm this message by clicking **Yes**.

The **TermStar database expert** window no longer displays the **TermStar NXT Received Access** database:

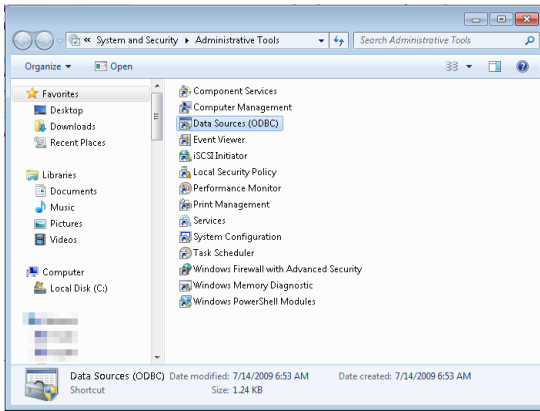


The old **TermStar NXT Access** database connection has been removed.

## Windows 7: Starting the 32-bit ODBC Data Source Administrator

Windows can be installed as a 32- or 64-bit operating system. In the 64-bit operating system, two versions of the ODBC Data Source Administrator are available (32- and 64-bit).

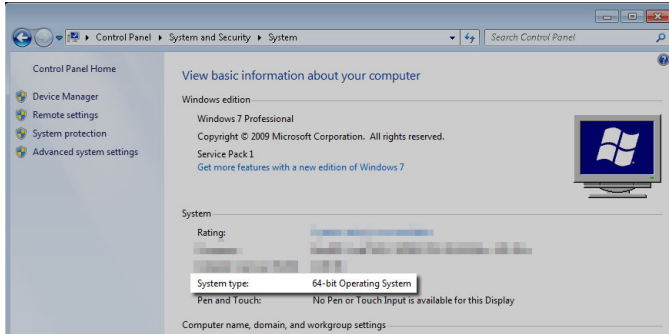
However, the 64-bit version of Windows displays only one version of the ODBC Data Source Administrator in the Control Panel:



In the Control Panel in Windows 7, you cannot tell whether the 32-bit or 64-bit version is linked.

For the administration of the TermStar data sources, the 32-bit version of the ODBC Data Source Administrator must be used for compatibility reasons.

You must therefore first determine whether you are working with a 32- or 64-bit operating system. To do so, select **System and Security | System** in the Windows Control Panel:



This window displays whether Windows is installed as a 64-bit operating system. If yes, you must manually start the ODBC Data Source Administrator in Windows 7.

- In the 32-bit operating system, only the 32-bit version of the ODBC Data Source Administrator is available. In this case, you can start the ODBC Data Source Administrator directly from the Control Panel.
- In the 64-bit operating system, both versions of the ODBC Data Source Administrator are available, but the Control Panel only displays the 64-bit version. In this case, you have to manually start the 32-bit version:
  - Open the `syswow64` Windows folder (the path is usually `C:\Windows\SystemWow64`).
  - Start the `odbcad32.exe` file as Administrator: Right-click the filename and select **Run as Administrator** in the context menu.

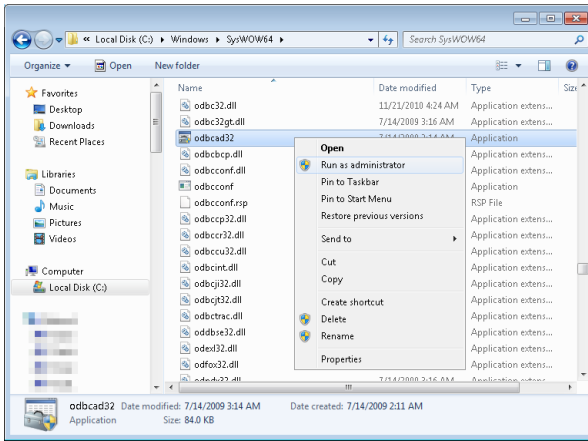


### Confusing folder and filenames in Windows

Do not let the folder and filenames in Windows confuse you:

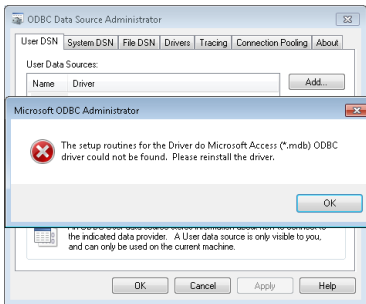
- The 32-bit version is in the `Windows/SystemWow64` folder.
- The 64-bit version is in the `Windows/System32` folder.

The filename for both versions is `odbcad32.exe`. The “32” in the filename therefore does not always mean that it is referring to the 32-bit version.



In Windows 7 with 64 bit, you must manually start the ODBC Data Source Administrator in the SysWOW64 folder.

If you have accidentally started the 64-bit version of the ODBC Data Source Administrator and attempt to configure Access databases from TermStar, Windows displays error messages:



With the 64-bit version, you cannot configure any Access databases from TermStar – the system will display error messages.

# 12 Fields in the TermStar dictionary

- What you should know here
- The TermStar data model consists of a predefined set of more than 10 entry types and over 40 entry fields, all of which can be individually adapted and used. The data model contains the following fields:
- Header fields ([» page 102](#))  
The header fields belong to a complete data record. They do not therefore relate to an individual language entry but to the overall semantic unit.
  - Language fields ([» page 103](#))  
The language fields relate to all language entries of one language. You can, for example, use these fields to create a single English description for several English language entries within a data record.
  - Language entry fields and subentry fields ([» page 104](#))  
The language entry and subentry fields contain the data for a language entry or subentry respectively. Each subentry you attach to an entry has the same number of fields as a main entry.

- Field formats
- TermStar supports the following formats of fields:
- Numerical field ("*Num.*"): Contains an automatically assigned number.
  - Date field ("*Date*"): Contains an automatically assigned number.
  - 255-character text field ("*255c*"): Contains up to 255 bytes of text.
  - 16-kB text field ("*16KB*"): Contains up to 16 kB of text.

## Header fields

Field name	Meaning	Format	Example
Data record number	Number assigned to each data record <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Num.	2905
GUID	Number which uniquely identifies each data record <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Num.	{2F16...EAC63}
External reference	External reference which uniquely identifies each data record <ul style="list-style-type: none"> <li>● Filled in by import or merging only</li> <li>● Cannot be edited</li> </ul>	16KB	LBP72336AM...22
Project	Project name that can be entered <u>by the user</u> (not filled automatically)	16KB	<i>Transit/TermStar Reference Guide</i>
Dictionary	Name of the dictionary that contains the data record <ul style="list-style-type: none"> <li>● Filled in automatically when the data record is created.</li> <li>● Cannot be edited</li> </ul>	255c	STAR
Status	Processing status of the data record	16KB	<i>Revised</i>
Multimedia	Linking of multimedia files	16KB	
Multimedia source	Source of the multimedia files	16KB	
Definition	Definition for all languages and language entries in this data record	16KB	
Definition source	Source of the definition	16KB	
User1 ... User9	Freely usable fields for the data record	16KB	
Created by	User who created the data record <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	<i>A. Smith</i>
Created on	Point in time when the data record was created <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	<i>29. May 2022, 16:30</i>
Last change by	User who last changed the data record <ul style="list-style-type: none"> <li>● Empty as long as the data record has not been changed</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	<i>A. Smith</i>

Header fields (meaning of Format column » [Field formats](#), page 101)

Field name	Meaning	Format	Example
Last change on	Point in time when the data record was last changed <ul style="list-style-type: none"> <li>● Empty as long as the data record has not been changed</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	4. December 2023, 09:36
Created or changed by	User who created the data record or who last changed it <ul style="list-style-type: none"> <li>● Never empty</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	A. Smith
Created or changed on	Point in time when the data record was created or last changed <ul style="list-style-type: none"> <li>● Never empty</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	4. December 2023, 09:36
Remark	General remarks on the entire data record	16KB	
Remark source	Source of the remarks	16KB	
Subject	Subject area of the data record	16KB	Data processing

Header fields (meaning of Format column » [Field formats](#), page 101) (cont.)

### Language fields

Field name	Meaning	Format	Example
Language	Language code (» <a href="#">Supported working languages</a> , page 142) <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	16KB	ENG
Info1 ... Info5	Freely usable fields that apply to all the language entries in this language	16KB	Styleguide 2016
Multimedia	Linking of multimedia files	16KB	
Multimedia source	Source of the multimedia files	16KB	
Definition	Definition for all language entries in this language in this data record	16KB	
Definition source	Source of the definition	16KB	

Language fields (meaning of Format column » [Field formats](#), page 101)

## 12 FIELDS IN THE TERMSTAR DICTIONARY

Language entry  
fields and  
subentry fields

Field name	Meaning	Format	Example
Entry number	Number assigned to each language entry <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Num.	195
GUID	Number which uniquely identifies each language entry <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Num.	{2F16...EAC63}
Term	Term of the language entry TermStar sorts the language entries according to this field content.	16KB	TermStar
Language	Language code of the language entry (» <b>Supported working languages</b> , page 142) <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	16KB	ENG
Created by	User who created the language entry <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	A. Smith
Created on	Point in time when the language entry was created <ul style="list-style-type: none"> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	29. May 2022, 16:30
Last change by	User who last changed the language entry <ul style="list-style-type: none"> <li>● Empty as long as the language entry has not been changed</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	A. Smith
Last change on	Point in time of the last change made to the language entry <ul style="list-style-type: none"> <li>● Empty as long as the language entry has not been changed</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	4. December 2023, 09:36
Created or changed by	User who created the language entry or who last changed it <ul style="list-style-type: none"> <li>● Never empty</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	255c	A. Smith

Entry fields and subentry fields (meaning of Format column » [Field formats](#), page 101)



Field name	Meaning	Format	Example
Created or changed on	Point in time when the language entry was created or last changed <ul style="list-style-type: none"> <li>● Never empty</li> <li>● Automatically filled in</li> <li>● Cannot be edited</li> </ul>	Date	4. December 2023, 09:36
Context	Context in which the language entry is used	16KB	<i>Terminology management</i>
Context source	Source of the context	16KB	<i>RefGuide</i>
Part of speech	Part of speech of the language entry	16KB	<i>Noun</i>
Gender	Gender of the language entry	16KB	<i>n.</i>
Grammatical information	Grammatical features of the language entry	16KB	<i>Proper name</i>
Status	Processing status of the language entry	16KB	<i>Revised</i>
Usage status	Usage status of the language entry	16KB	<i>Preferred</i>
Data source	Source of the language entry	16KB	<i>Terminology Department</i>
Attributes	Attributes for the language entry	16KB	<i>New</i>
Remark	General remarks on the language entry	16KB	<i>Note the spelling!</i>
Remark source	Source of the remarks	16KB	
Subject	Subject area of the language entry	16KB	<i>Translation, software</i>
Phonetic information	Information on the pronunciation of the term	16KB	
Cross-reference	Field for cross-reference <p>Examples:</p> <ul style="list-style-type: none"> <li>● Cross-reference to a related language entry</li> <li>● Cross-reference to an internet address</li> </ul>	16KB	<i>Transit NXT</i>
Definition	Definition of the language entry	16KB	<i>STAR's Translation Memory Tool</i>
Definition source	Source of the definition	16KB	
User1 ... User9	Freely usable fields for the language entry	16KB	

Entry fields and subentry fields (meaning of Format column » [Field formats](#), page 101) (cont.)

Prefixes for field types In selection lists and in the layout editor (» [Working with the layout editor](#), page 43), TermStar displays a field name prefix for each field type:

Field type	Prefix
Header field	Hdr.
Language field	Lang.
Language entry field	Entr.

Prefix for field names in selection lists

Field type	Prefix
Subentry field:	
● Abbreviation	Abbr.
● Synonym	Syn.
● Alternative	Alt.
● Irregular form	Irreg.
● Disallowed term	Dis. term.
● User index 1	U. ind.1
● User index 2	U. ind.2
● User index 3	U. ind.3
● User index 4	U. ind.4
● User index 5	U. ind.5

Prefix for field names in selection lists (cont.)

# 13 Regular expressions

**What are regular expressions?** Regular expressions are used to define character strings that match a specific pattern. You can use them in any situation where you want Transit or TermStar to search for and/or replace items of text.

That gives you much flexibility so that you can even perform complex Find and Replace sequences in a single operation. When performing a normal search without using regular expressions, you can only specify a set search phrase.

- **Example:**

You want to find all occurrences of the expressions `Year 2015`, `Year 2016` and `Year 2017`.

If you enter the search phrase `Year 2016` in the Transit editor, Transit will find each occurrence of precisely the phrase `Year 2016`. With a standard search, therefore, you would have to run three separate searches for the phrases `Year 2015`, `Year 2016` and `Year 2017` respectively.

However, if you entered the regular expression `Year 201[5-7]`, Transit would find all three phrases in a single search operation. The precise meaning of the expression is explained later on ([\\* Wildcard for any of a specified group or class: \[ \]](#), page 114).



### **Transit and TermStar support regular expressions**

Wherever this chapter describes a regular expression for use in Transit, you can also use it in TermStar. Syntax and function are for the most part identical.

If a distinction between Transit and TermStar is necessary, it is explicitly indicated.

**Basic settings for searches in Transit** Unless you change them, Transit uses the following basic settings for searches:

- **No distinction between upper and lower case**

By default Transit does not distinguish between upper and lower case.

Example: Searching for `STAR` will also find `Star`, `star` and `stAr`.

To force the regular expression to find only those occurrences that match the case of your search string, you must select the option **Match case** in Transit.

- **No distinction between whole words and parts of words**

By default Transit does not distinguish between whole words and parts of words.

Example: Searching for `star` will also find `Elstar`, `start` and `starring`.

To force the regular expression to find whole words only, you must select the option **Find whole words** only in Transit.

- Regular expressions not recognised

By default, Transit interprets search strings literally rather than as regular expressions.

To force Transit to interpret a search string as a regular expression, you must select the option **Regular expression**.

What can you use regular expressions for? You can use regular expressions in the following functions:

- In Transit
  - Find
  - Find/Replace
  - Segment Filter
  - Translation exceptions
  - File type definition (tag definition and protection; segmentation)
- In TermStar
  - Data record filter
  - Input verification
  - Data manipulation (Find/Replace)
  - Dictionary import (preprocessing, field definitions in expert mode, substitutions)

## Overview of meta and control characters

The table below summarises the meta and control characters used in regular expressions. For a more detailed description of the individual characters, refer to sections quoted.

	Meta characters	Meaning
Control characters	\b	Backspace
	\e	Escape (ESC)
	\f	Form feed (new page)
	\n	New line
	\o	Segment marker (in Transit editor only)
	\s	Space
	\t	Tabulator
	\u<code>	Unicode character
	The backslash is also a meta character for an escapement (» <a href="#">Escapement: \</a> , page 121).	
Wildcard/character class	Dot: .	Wildcard for any single character » <a href="#">Wildcard for any single character: . (dot)</a> , page 114.
	Square brackets: [xyz]	Character class » <a href="#">Wildcard for any of a specified group or class: [ ]</a> , page 114
	Ampersand: &	Wildcard for any sequence of characters » <a href="#">Wildcard for any sequence of characters: &amp;</a> , page 116
Quantifier	Question mark: x?	Finds occurrences of absence or a single instance (0 – 1 instances) of the preceding character » <a href="#">Quantifiers: + * ?</a> , page 118
	Plus sign: x+	Finds occurrences of a single instance or multiple instances (1 – n instances) of the preceding character. » <a href="#">Quantifiers: + * ?</a> , page 118
	Asterisk: x*	Finds occurrences of the absence, a single instance or multiple instances (0 – n instances) of the preceding character. » <a href="#">Quantifiers: + * ?</a> , page 118
Escapement	Backslash: \x	Forces character to be interpreted literally instead of as a meta character. » <a href="#">Escapement: \</a> , page 121
	The backslash also introduces control characters (» <a href="#">Control characters</a> , page 112).	
Structural	Round brackets: (xyz)	You can use round brackets to structure a regular expression to determine, for example, for which part of the expression a preceding meta character should apply. » <a href="#">Applying meta characters to character strings: ( )</a> , page 123.

Meta characters and control characters used in regular expressions

	Meta characters	Meaning
Logical operator	Circumflex: ^	Beginning of line or field: The expression must occur at the beginning of a line or language entry field.  » Placement: ^ \$, page 124
	Dollar sign: \$	End of line or field: The expression must occur at the end of a line or language entry field.  » Placement: ^ \$, page 124
	Exclamation mark: (!xxx) [!x]	Negation: The expression which follows may <u>not</u> occur.  » Negation: !, page 126
	Pipe character: Expression1 Expression2	Alternative: Combines two expressions with a logical OR operation. Finds expression1 <u>or</u> expression2.  » Alternatives:  , page 129
Variable	#(…)<n1r>	Variable (in search): Stores character string for use in replacement.  » Variables: #, page 132
	#<n1r>	Variable (in replacement): Inserts the variable stored by the search operation in the replacement.  » Variables: #, page 132

Meta characters and control characters used in regular expressions (cont.)

## Defining regular expressions

Regular expressions are made up of the following components:

- Standard characters

Transit searches for the standard characters as they appear in the regular expression. They have no special meaning.

The standard characters are letters, numbers and special characters excepting those characters that serve as meta characters (`.`, `&`, `*`, `+`, `?`, `[`, `]`, `(`, `)`, `$`, `^`, `!`, `\`, `|`, `#`). More details of the meta characters are given later on.

Example:

- The character `a` in a regular expression will find the character `a`.
- The character string `star-group` in a regular expression will find the character string `star-group`.

- Control characters (» [page 112](#))

Control characters are non-printing characters that control the appearance of the text (e.g. tabulator, line break, etc.). Control characters can also be used to search for Unicode characters or replace characters with Unicode characters.

- Meta characters

Meta characters have special meanings when used in regular expressions. They are used to define the pattern that Transit searches for.

Transit treats the following characters as meta characters:

`.` `&` `*` `+` `?` `[` `]` `(` `)` `$` `^` `!` `\` `|` `#`

The meanings of the individual meta characters are explained in section » [Overview of meta characters](#), page 113.

## Control characters

Control characters are non-printable characters that control the appearance of the text (e.g. tabulator, line break, etc.).

Control characters can also be used to search for Unicode characters or replace characters with Unicode characters.

Control characters are introduced by a backslash (\).

	Meta characters	Meaning
Control characters	<code>\b</code>	Backspace
	<code>\e</code>	Escape (ESC)
	<code>\f</code>	Form feed (new page)
	<code>\n</code>	New line Notes: <ul style="list-style-type: none"> <li>• The control character <code>\n</code> finds line breaks <u>inside</u> segments. It does <u>not</u> find line breaks created by segment markers (e.g. new paragraphs). To find such line breaks you can use the control character <code>\o</code> (for segment markers).</li> <li>• The expression <code>\n</code> is used to find the end of a line. The characters <code>^</code> and <code>\$</code> are used to search for the search string at the beginning and end of a line respectively. In many cases both expressions return the same result but there can be differences both in usage and outcome (» <a href="#">Control character <code>\n</code> versus placement characters <code>^</code> and <code>\$</code></a>, page 125).</li> </ul>
	<code>\o</code>	Segment marker (in Transit editor only)
	<code>\s</code>	Space
Unicode characters	<code>\t</code>	Tabulator
	<code>\u&lt;code&gt;</code>	Unicode characters <code>&lt;code&gt;</code> is the hexadecimal code for the Unicode character. Examples: <ul style="list-style-type: none"> <li>• <code>\u20AC</code>: Unicode character 20AC (Euro symbol)</li> <li>• <code>\u0394</code>: Unicode character 394 (character Δ)</li> </ul>



### Backslash also meta character for escapement

The backslash not only introduces control characters, but it also serves as a meta character for an escapement so that a meta character can be searched for in its literal form (» [Escapement: `\`](#), page 121).



### Finding/replacing control characters without using regular expressions

You can also use control characters in Find and Replace operations without regular expressions. If you do not wish to use meta characters in a Find/Replace operation, you can deselect the option **Regular expressions**. Transit will still find the control characters.



## Overview of meta characters

Meta characters have special meanings when used in regular expressions. They are used to define the pattern that Transit searches for. Transit recognises the following types of meta character:

- Wildcards: `.` `[]` & (» [page 114](#))  
Wildcards are characters that are used to represent any single character or sequence of characters.
- Quantifiers: `+` `*` `?` (» [page 118](#))  
By using what are known as *quantifiers* you can specify how many instances of a character are to be found.
- Escapement: `\` (» [page 121](#))  
If you wish to search for a meta character literally (in other words treat it as a standard character and not as having a special meaning) you must place a backslash before it.
- Applying meta characters to character strings: `()` (» [page 123](#))  
By using round brackets `()` in a regular expression, you can specify that a meta character applies to a character string rather than a single character.
- Placement: `^` `$` (» [page 124](#))  
The meta characters that define placement are used to specify whether the character string is placed at the beginning or end of a line (in Transit) or language entry field (in TermStar).
- Negation: `!` (» [page 126](#))  
The exclamation mark `!` is used to negate part of a regular expression. In that way you can instruct Transit to find characters that do not match the negated part of the expression.
- Alternatives: `|` (» [page 129](#))  
The pipe character `|` allows you to search for alternatives. The pipe character acts as a logical OR between parts of a regular expression.
- Variables: `#` (» [page 132](#))  
You can use variables in a Find and Replace operation to define variable components of the character string that Transit is to search for. That allows you to perform complex Find and Replace sequences in a single operation.

## Wildcards: . [ ] &

Wildcards are characters that are used to represent any single character or sequence of characters.

**Wildcard for any single character:** You can use a dot (.) to search for any single character.  
 . (dot)

“Any character” means any letter (including letters with accents), number, special character (e.g. @, %, \_), meta characters (e.g. \$, . or &), space character, control character (e.g. tabulator) or double-byte character. The only character that Transit does not find is the control character for a line break.

Regular expression	Matches	Does not match
st.r	<ul style="list-style-type: none"> <li>● STAR</li> <li>● stir</li> <li>● stër</li> <li>● stör</li> </ul>	<ul style="list-style-type: none"> <li>● STAAR</li> <li>● st r</li> </ul>
On.line	<ul style="list-style-type: none"> <li>● On-line</li> <li>● On line</li> <li>● On..line</li> </ul>	<ul style="list-style-type: none"> <li>● Online</li> </ul>

Wildcard for any single character

**Wildcard for any of a specified group or class:** [ ] If you want Transit to search for any one of a particular group of characters you can define a character group or class. To do this, place the valid characters in square brackets [ ]. You can specify character groups, classes or a combination of both.

- Character group: A group of multiple individual characters  
 Example: [aeiou]: This group consists of all of the vowels
- Character class: A contiguous range of characters  
 This is a simpler way of specifying a group of consecutive characters rather than listing each character in a group.

Examples:

- [a-k]: All letters from a to k inclusive (equivalent to character group [abcdefghijklmnop])
- [3-5]: The numbers 3, 4 and 5 (equivalent to character group [345])
- [\u3349-\u4221]: Unicode characters 3349 to 4221

To define a character class, you specify the first and last characters of a consecutive group. Transit treats all characters that are between the specified characters in the ANSI or Unicode table as belonging to the character class.

- Combination of character groups and classes  
 You can combine character groups and classes inside the square brackets.  
 Example:

[a-zäöü]: Letters a to z plus German umlaut characters

Regular expression	Matches	Does not match
st[aeiou]r	<ul style="list-style-type: none"> <li>● star</li> <li>● stir</li> <li>● stor</li> </ul>	<ul style="list-style-type: none"> <li>● stör</li> <li>● strr</li> <li>● STAAR</li> </ul>
201[3-7]	<ul style="list-style-type: none"> <li>● 2015</li> <li>● 2016</li> <li>● 2017</li> </ul>	<ul style="list-style-type: none"> <li>● 1915</li> <li>● 2012</li> <li>● 2035</li> </ul>
st[a-zäöü]r	<ul style="list-style-type: none"> <li>● star</li> <li>● stör</li> <li>● strr</li> </ul>	<ul style="list-style-type: none"> <li>● stár</li> <li>● STAAR</li> <li>● st@r</li> </ul>
[a-z][0-9]	<ul style="list-style-type: none"> <li>● a4</li> <li>● k3</li> <li>● z5</li> </ul>	<ul style="list-style-type: none"> <li>● 1a</li> <li>● mx</li> <li>● 68</li> </ul>

Wildcard with character group or class



### Negation of a character group

You can specify that a character group does not include specific characters. This significantly simplifies the definition of character groups (» [Negation of a character group](#), page 127).



### Use of case in character group/class definitions

When defining character groups/classes remember to take account of the Match case option setting in Transit.

Example:

The character class [a-d] can signify either of the following:

- If the Match case option is not selected: Upper and lower-case letters from a to d (i.e. a, b, c, d, A, B, C, D).
- If the Match case option is selected: Only all the lower-case letters from a to d (i.e. a, b, c, d).

If the Match case option is selected, and you want to find all letters in the character class regardless of case, you must enter upper and lower-case letters in the class definition (i.e. [a-dA-D]).



### Meta characters within square brackets

Within square brackets, almost all meta characters lose their special meanings.

The following characters have a special meaning even within the square brackets:

- With **!** at the start of a character group you can specify which characters are not permitted (» [Negation of a character group](#), page 127).
- A hyphen (-) placed between two characters specifies a range.

Example: `[a-kA-K]` applies to all character strings with upper and lower-case letters from A to K.



### Umlauts, accented letters, etc.

**The character class `[a-z]` does not include letters with umlauts, accents, etc.**

If you wish to search for character groups that include such characters, you must add those characters to the class definition (e.g. `[a-zäöüß]` for the German alphabet).

Wildcard for any sequence of characters: `&`

You can use the ampersand (`&`) to represent any sequence of characters. It is used to define an unlimited sequence of any characters bounded by delimiters.

In this case, you must always specify the characters by which the sequence is bounded (the “*beginning delimiter*” and the “*end delimiter*”).

- Example: `s&r` finds `s` followed by any combination of any number of characters followed by `r`.

“Any character” means any letter (including letters with accents), number, special character (e.g. `ø`, `%, _`), meta characters (e.g. `$`, `.` or `&`), space character, control character (e.g. tabulator) or double-byte character.

Transit only finds occurrences where the beginning and end delimiters are in the same segment. If there is a segment marker between the beginning and end delimiters, Transit does not find the search string.

- Example: `s&r` finds `s` followed by any combination of any number of characters followed by `r`.
  - In the following two segments, Transit does not find the expression because the beginning and end delimiters are not in the same segment:
 

```
solid<<29>>
as a rock<<30>>
```
  - If, however, the beginning and end delimiters are in the same segment Transit finds the character string:
 

```
solid as a rock<<29>>
```

Regular expression	Matches	Does not match
s&r	<ul style="list-style-type: none"> <li>● STAR</li> <li>● Saint Peter</li> <li>● S?.!R</li> <li>● SR</li> <li>● s r (line break)</li> </ul>	<ul style="list-style-type: none"> <li>● mar (beginning delimiter s is missing)</li> <li>● stone (end delimiter r is missing)</li> <li>● solid&lt;&lt;29&gt;&gt; as a rock&lt;&lt;30&gt;&gt; (segment marker between beginning and end delimiters)</li> </ul>

Wildcard for any sequence of characters:



### Do not forget beginning and end delimiters!

**Without beginning or end delimiter, the regular expression is invalid.**

The following regular expressions are incorrect: s& (end delimiter missing) and &r (beginning delimiter missing).

When you enter a regular expression using the ampersand, Transit searches as follows:

- Transit first searches for the beginning delimiter. Transit starts highlighting/ marking the text from the first occurrence found.
- Transit then searches for the end delimiter while continuously extending the highlight.
- As soon as Transit finds the end delimiter, it highlights it and stops searching.

Example:

- You search for the regular expression s&r in the following passage of text:  
The stars are shining bright.
- The passage contains three consecutive occurrences of the specified regular expression.
  - The stars are shining bright.
  - The stars are shining bright.
  - The stars are shining bright.
- If you perform the search three times in succession, Transit will thus find the following three character strings:
  - star
  - s ar
  - shining br

More examples are given in section » [Quantifiers for character classes and groups](#), page 119.



### Distinction between ampersand (&) and the expression .\*

You can use the ampersand (&) to represent any sequence of characters. The expression .\* is used to find any single or multiple occurrence of any character or no character.

In many cases both expressions return the same result but there can be differences both in usage and outcome (» [Distinction between ampersand \(&\) and the expression .\\*](#), page 120).

## Quantifiers: + \* ?

By using what are known as *quantifiers* you can specify how many instances of a character are to be found.

- Question mark *x?*  
Finds occurrences of absence or a single instance (0 – 1 instances) of the preceding character.  
Example: The regular expression *a?* finds a single letter *a* or the absence of it.
- Plus sign *x+*  
Finds occurrences of a single instance or multiple instances (1 – *n* instances) of the preceding character.  
Example: The regular expression *a+* finds a single letter *a* or multiple sequences of it (*a*, *aa*, *aaa* etc.).
- Asterisk *x\**  
Finds occurrences of the absence, a single instance or multiple instances (0 – *n* instances) of the preceding character.  
Example: The regular expression *a\** finds a single letter *a*, a multiple sequence of it, or its absence (*a*, *aa*, *aaa* etc. or nothing).

When searching, Transit only searches as far as a segment marker or line break and extends the highlight up to the last character preceding the segment marker or line break.

- Example: The regular expression *a\** searches for the absence of the letter *a*, a single instance of it or a multiple sequence of it.

In the following two lines, Transit first finds only the string *aa* as it is followed by a line break.

```
baa
as sheep do
```

On continuing the search, Transit finds the single *a* on the next line.

baa  
as sheep do

Regular expression	Matches	Does not match
sta?r	<ul style="list-style-type: none"> <li>● str</li> <li>● star</li> </ul>	<ul style="list-style-type: none"> <li>● staaaar</li> <li>● stra</li> <li>● stir</li> </ul>
sta+r	<ul style="list-style-type: none"> <li>● star</li> <li>● staaaar</li> </ul>	<ul style="list-style-type: none"> <li>● str</li> <li>● stra</li> <li>● stir</li> <li>● staa aar (only searches as far as line break)</li> </ul>
sta*r	<ul style="list-style-type: none"> <li>● str</li> <li>● star</li> <li>● staaaar</li> </ul>	<ul style="list-style-type: none"> <li>● stra</li> <li>● stir</li> <li>● staa aar (only searches as far as line break)</li> </ul>

Searching for a specified number of a specific character

You can also use quantifiers to specify how many instances of a character in a character group or class are to be found. To do so, you place the quantifier after the square brackets in which the character group or class is defined (» [Wildcard for any of a specified group or class: \[ \]](#), page 114).

- Example: The regular expression `[0-9]+` finds any sequence of numbers (0, 15, 290504 etc.).

Regular expression	Matches	Does not match
<code>[0-9]+</code>	<ul style="list-style-type: none"> <li>● 0</li> </ul>	<ul style="list-style-type: none"> <li>● xV</li> </ul>
Finds sequences of one or more numbers. Other characters are not found.	<ul style="list-style-type: none"> <li>● 15</li> <li>● 290504</li> </ul>	<ul style="list-style-type: none"> <li>● star</li> </ul>
<code>st[aeiou]*r</code>	<ul style="list-style-type: none"> <li>● str</li> </ul>	<ul style="list-style-type: none"> <li>● stör</li> </ul>
Finds sequences of one or more vowels or absence of vowels between the delimiters. Other characters are not found.	<ul style="list-style-type: none"> <li>● star</li> <li>● stir</li> <li>● staaaar</li> <li>● stair</li> </ul>	<ul style="list-style-type: none"> <li>● stpr</li> <li>● st@r</li> </ul>
<code>[A-Z][a-z]*</code>	<ul style="list-style-type: none"> <li>● D</li> </ul>	<ul style="list-style-type: none"> <li>● translation</li> </ul>
Finds any single upper-case letter followed by any sequence of (or absence of) lower-case letters. Other characters are not found.	<ul style="list-style-type: none"> <li>● Star</li> <li>● Transit</li> </ul>	<ul style="list-style-type: none"> <li>● Übersetzung</li> <li>● TermStar</li> </ul>

Quantifiers for character classes and groups

Regular expression	Matches	Does not match
[A-Z][a-z]+	<ul style="list-style-type: none"> <li>● Star</li> </ul>	<ul style="list-style-type: none"> <li>● D</li> </ul>
Finds strings where at least one lower-case letter follows an upper-case letter. Other characters are not found.	<ul style="list-style-type: none"> <li>● Transit</li> </ul>	<ul style="list-style-type: none"> <li>● translation</li> <li>● Übersetzung</li> <li>● TermStar</li> </ul>
Star[a-z]?	<ul style="list-style-type: none"> <li>● Star</li> </ul>	<ul style="list-style-type: none"> <li>● Star3</li> </ul>
Finds strings where the literal string is followed by nothing or a single letter. Other characters are not found.	<ul style="list-style-type: none"> <li>● Start</li> <li>● Stars</li> </ul>	<ul style="list-style-type: none"> <li>● Starter</li> </ul>

Quantifiers for character classes and groups (cont.)



### Use of case in character group/class definitions

When defining character groups/classes remember to take account of the Match case option setting in Transit (» [Use of case in character group/class definitions](#), page 115).



### Quantifiers for character strings

Normally, a quantifier applies to the character that immediately precedes it.

With round brackets ( ), you can specify that the quantifier applies to a character string rather than a single character (» [Applying meta characters to character strings: \( \)](#), page 123).



### Distinction between ampersand (&) and the expression .\*

You can use the ampersand (&) to represent any sequence of characters. The expression .\* is used to find any single or multiple occurrence of any character or no character.

In many cases both expressions search the same but there can be differences both in usage and outcome:

- & requires beginning and end delimiters.  
By contrast, the expression .\* can also be used without specifying a beginning or end delimiter.
- & also finds line breaks within a segment.  
By contrast, the expression .\* does not find line breaks and highlights the character string up to the last character preceding the line break.

For a more detailed description of using the ampersand, refer to section » [Wildcard for any sequence of characters: &](#), page 116.



## Escapement: \

Meta characters have special meanings when used in regular expressions. They are used to define the pattern that Transit searches for.

If you wish to search for a meta character literally (in other words treat it as a standard character and not as having a special meaning) you must place the character \ (backslash) before it. The backslash acts as what is called an “*escapement*” and prevents the meta character from being interpreted as having a special meaning.

The same applies to the backslash itself, as it too is a meta character. If you want to search literally for a backslash, you have to place another backslash before it.

- Examples:
  - You want to search for the string `readme.txt`.  
The dot, however, is a meta character (» [Wildcard for any single character: . \(dot\)](#), page 114).  
So that Transit searches literally for a dot, you have to place a backslash before it thus: `readme\.txt`
  - You want to search for the string `Transit+TermStar`.  
However, the plus sign is a meta character (» [Quantifiers: + \\* ?](#), page 118).  
So that Transit searches literally for a plus sign, you have to place a backslash before it thus: `Transit\+TermStar`
  - You want to search for the string `folder \temp`.  
However, the backslash is a meta character (escapement) and also introduces control characters (e.g. the tabulator \t).  
So that Transit searches literally for a backslash, you have to place another backslash before it thus: `folder \\temp`

The following table shows the use of the escapement in regular expressions for literal searches for meta characters:

Regular expression	Matches	Does not match
<code>readme\.txt</code>	● <code>readme.txt</code>	● <code>readmextxt</code> ● <code>readme-txt</code> ● <code>readmetxt</code>
<code>Transit\+TermStar</code>	● <code>Transit+TermStar</code>	● <code>TransitTermStar</code> ● <code>TransitTermStar</code> ● <code>TransittttttTermStar</code>
<code>folder \\temp</code>	● <code>folder \temp</code>	● <code>folder temp</code> ● <code>folder</code> <code>emp</code>

Literal searches for meta characters using escapement

The same expressions without the escapement do not produce the desired results:

Regular expression	Matches	Does not match
<code>readme.txt</code>	<ul style="list-style-type: none"> <li>● <code>readme.txt</code></li> <li>● <code>readmextxt</code></li> <li>● <code>readme-txt</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>readmetxt</code></li> </ul>
<code>Transit+TermStar</code>	<ul style="list-style-type: none"> <li>● <code>TransitTermStar</code></li> <li>● <code>TransitttttTermStar</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>Transit+TermStar</code></li> <li>● <code>TransiTermStar</code></li> </ul>
<code>folder \temp</code>	<ul style="list-style-type: none"> <li>● <code>folder emp</code> (\t is the control character for the tabulator)</li> </ul>	<ul style="list-style-type: none"> <li>● <code>folder \temp</code></li> <li>● <code>folder temp</code></li> </ul>

Incorrect results produced by literal searches for meta characters without using escapement



### Backslash also introduces control characters

The backslash not only serves as a meta character for an escapement, but it also introduces control characters (» [Control characters](#), page 112).

## Applying meta characters to character strings: ( )

With round brackets ( ), you can specify that a meta character applies to a character string rather than a single character.

- Example: The plus sign specifies that the preceding character should occur once or more than once (» [Quantifiers: + \\* ?](#), page 118). If you enclose a character string in round brackets, the plus sign applies to the entire character string thus: The regular expression (ha)+ finds sequences in which the string ha occurs once or any number of times (ha, haha, hahaha, etc.).

By combining character strings, character groups and quantifiers, you can define complex patterns as illustrated below:

Regular expression	Matches	Does not match
s(ta)+r	● star	● sr
s, followed by one or more occurrences of ta, followed by r	● statar ● statatatar	● stat ● statr
T(ra)+[a-z]+	● Transit	● Termstar
T, followed by one or more occurrences of ra, followed by any single letter or sequence of letters	● Traransit ● Transfer ● Trararas	● Tra2004 ● Tra-nsit

Effect of round brackets on meta characters



### Every open bracket requires a closing bracket

**Make sure that for every opening bracket in the regular expression there is a corresponding closing bracket.**

The regular expression is otherwise invalid.



### Tip: Use round brackets to structure the expression

You can also use round brackets to structure regular expressions and make them clearer – even if use of the brackets is not strictly necessary for syntax reasons.

In addition, round brackets are also required when using the following meta characters:

- Negation: ! (» [page 126](#))
- Alternatives: | (» [page 129](#))
- Variables: # (» [page 132](#))

## Placement: ^ \$

The meta characters that define placement are used to specify whether the character string is placed at the beginning or end of a line (in Transit) or language entry field (in TermStar).

Transit checks whether the search string is at the beginning or end but does not highlight the beginning or end itself.

- Circumflex ^

By using the circumflex you can specify that the search string must be placed at the beginning:

- In Transit: At the beginning of a line (i.e. after a line break)
- In TermStar: At the beginning of a language entry field

To do so, you place the circumflex at the beginning of the regular expression.

- Example: The regular expression `^STAR` searches for occurrences of `STAR` at the beginning of a line.

- Dollar sign \$

By using the dollar sign you can specify that the search string must be placed at the end:

- In Transit: At the end of a line (i.e. before a line break)
- In TermStar: At the end of a language entry field

To do so, you place the dollar sign at the end of the regular expression.

- Example: The regular expression `STAR$` searches for occurrences of `STAR` at the end of a line.

Regular expression	Matches	Does not match
<code>^STAR</code>	<ul style="list-style-type: none"> <li>● STAR in STAR Group...</li> </ul>	<ul style="list-style-type: none"> <li>● STAR in The STAR Group...</li> </ul>
<code>STAR\$</code>	<ul style="list-style-type: none"> <li>● STAR in ... with STAR</li> </ul>	<ul style="list-style-type: none"> <li>● STAR in ... STAR Group</li> <li>● STAR in ... STAR.</li> </ul>
<code>^&amp;\$</code> (Any sequence of characters between beginning and end of line)	<ul style="list-style-type: none"> <li>● The STAR Group in The STAR Group</li> </ul>	<ul style="list-style-type: none"> <li>● The STAR group (spans more than one line)</li> </ul>

Regular expression at the start of a line



### Transit editor: Line break is not the same as segment marker

In the Transit editor the meta characters for placement check whether the search string is before or after a line break, i.e. before or after the control character `\n`.

It does not find line breaks created by segment markers (e.g. new paragraphs). To find such line breaks you can use the control character `\o` (for segment markers).



### Correct placement of meta characters

Take care to place the meta characters correctly within the regular expression:

- `^` (for beginning of line) at the beginning of the regular expression.
- `$` (for end of line) at the end of the regular expression.

Otherwise the regular expression will not be correct, and Transit will not be able to interpret it properly.



### Negation of beginning/end of line

You can also specify that Transit is to search for occurrences of the regular expression that are not at the beginning/end of a line. To do so, you use the exclamation mark to negate the beginning/end of the line (» [Negation of beginning/end of line](#), page 128).



### `^` and `$` within character groups/classes are no meta characters

Inside a character group, Transit interprets the circumflex and dollar sign literally, i.e. searches for the circumflex and dollar sign characters themselves (» [Wildcard for any of a specified group or class: \[ \]](#), page 114).



### Control character `\n` versus placement characters `^` and `$`

The expression `\n` is used to find the end of a line. The characters `^` and `$` are used to search for the search string at the beginning and end of a line respectively.

In many cases both expressions search the same but there can be differences both in usage and outcome:

- `^` and `$` must be placed at the beginning and end of the regular expression respectively.

By contrast, the control character `\n` can also be placed within the regular expression. In that way you can search for character strings that span line breaks.

- `^` and `$` do not find the line break itself.

By contrast, the control character `\n` finds and highlights the line break itself. In that way you find and replace the line break if you wish.

## Negation: !

The exclamation mark (!) is used to negate part of a regular expression. In that way you can instruct Transit to find characters that do not match the negated part of the expression.



### Difference in highlighting between negation of character string and character group/class

Transit behaves differently when characters and character strings are negated than when character groups or classes are negated.

- When characters or character strings are negated, Transit does not highlight the negated character.
- When character groups or classes are negated, Transit also highlights the negated character.

For examples, refer to the Tables » [Negation of character strings](#), page 126 and » [Negation of character groups](#), page 127.

### Negation of a character or character string

You can specify that Transit is to search for sequences that do not include a specific character or character string.

To do so, you place the exclamation mark and the character or characters inside round brackets.

- Example: The regular expression `ST (!ONE)` finds any occurrence of the string `ST` that is not followed by the string `ONE`.

Regular expression	Matches	Does not match
<code>ST (!ONE)</code>	<ul style="list-style-type: none"> <li>● <code>ST</code> in <code>STAR</code></li> <li>● <code>ST</code> in <code>STIR</code></li> <li>● <code>ST</code> in <code>ST</code></li> <li>● <code>ST</code> in <code>STP</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>ST</code> in <code>STONE</code></li> </ul>

Negation of character strings

Transit only interprets the exclamation mark as a meta character for negation if it is placed as the first character inside the round brackets.

Otherwise, Transit interprets the exclamation mark literally, i.e. it searches for the exclamation mark itself:

Regular expression	Matches	Does not match
ST(ON!E)	● STON!E	● STONE ● STAR ● STIR ● ST ● STP

If the exclamation mark is not the first character, it is interpreted literally

### Negation of a character group

You can specify that a group of characters is not included. This significantly simplifies the definition of character groups (» [Wildcard for any of a specified group or class: \[ \]](#), page 114).

To negate a character group you place the exclamation mark as the first character of the character group inside the square brackets.

- Example: You want to define a character group that includes all characters except `s`. Without using negation, you would have to specify all characters in the group:

```
[a-rt-z0-9ß?-=@]
```

By using negation, the definition is much simpler, and you have greater certainty of including all the required characters:

```
[!s]
```

- Example: The regular expression `ST[!ONE]` finds any character string that consists of three characters, starts with `ST` and does not end in `O`, `N` or `E`.

Regular expression	Matches	Does not match
ST[!ONE]	● STA in STAR  ● STP in STP	● STO in STONE  ● STE in STELLA  ● ST in ST

Negation of character groups

Transit only interprets the exclamation mark as a meta character for negation if it is placed as the first character inside the square brackets.

Otherwise, Transit interprets the exclamation mark literally, i.e. it searches for the exclamation mark itself:

Regular expression	Matches	Does not match
ST(ON!E)	● STO ● STN ● ST! ● STE	● STA ● ST5

If the exclamation mark is not the first character, it is interpreted literally

**Negation of beginning/end of line** By negating the meta characters that define placement (circumflex and dollar sign) you can specify that Transit is to search for occurrences of the character string that are not at the beginning or end of a line (» **Placement: ^ \$**, page 124).

To do so, you place the exclamation mark and the circumflex or dollar sign inside round brackets.

- Examples:
  - The regular expression `(!^)STAR` searches for occurrences of `STAR` that are not at the beginning of a line.
  - The regular expression `STAR(!$)` searches for occurrences of `STAR` that are not at the end of a line.

Regular expression	Matches	Does not match
<code>(!^)STAR</code>	<ul style="list-style-type: none"> <li>● <code>STAR</code> in <code>The STAR Group...</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>STAR</code> in <code>STAR Group...</code></li> </ul>
<code>STAR(!\$)</code>	<ul style="list-style-type: none"> <li>● <code>STAR</code> in ... <code>STAR Group</code></li> <li>● <code>STAR</code> in ... <code>STAR.</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>STAR</code> in ... with <code>STAR</code></li> </ul>

Negation of beginning or end of line

If you only want to negate the beginning/end of line placement, take care to place the appropriate meta character in round brackets on its own. Otherwise the entire expression would be negated.

- Example:
  - The regular expression `ST(!AR$)` finds any occurrence of the string `ST` that is not followed by the string `AR` and the end of a line.

Regular expression	Matches	Does not match
<code>ST(!AR\$)</code> Negation of entire string <code>AR</code> plus end of line	<ul style="list-style-type: none"> <li>● <code>ST</code> in ... <code>STAR Group</code></li> <li>● <code>ST</code> in with <code>ST</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>ST</code> in with <code>STAR</code></li> </ul>

Negation of entire string `AR` plus end of line

Transit only interprets the exclamation mark as a meta character for negation if it is placed as the first character of the character string inside the round brackets.

Otherwise, Transit interprets the exclamation mark literally, i.e. it searches for the exclamation mark itself:

Regular expression	Matches	Does not match
<code>ST(AR!\$)</code>	<ul style="list-style-type: none"> <li>● <code>STAR!</code> in ... <code>STAR!</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>STAR</code> in ... with <code>STAR</code></li> <li>● <code>STAR!</code> in ... <code>STAR! Group</code></li> </ul>

If the exclamation mark is not the first character, it is interpreted literally



## Alternatives: |

The pipe character `|` allows you to search for alternatives. The pipe character acts as a logical OR between parts of a regular expression.

- Example: The regular expression `Transit|TermStar` finds either the character string `Transit` or `TermStar`.

`Transit` finds the first match with either of the two alternatives. It makes no difference in which order the alternatives are specified in the regular expression.

- Example: In the passage `TermStar` and `Transit` are using regular expressions. `Transit` will first find the string `TermStar` because it is the first occurrence of either of the two alternatives specified in the regular expression.

What does `Transit` interpret as an alternative?

By default, `Transit` interprets everything from the beginning of the regular expression to the first pipe character, between two pipe characters or from the last pipe character to the end of the regular expression as an alternative.

- Example: `Transit` interprets the regular expression `You can use the English|German|Swedish interface` as the following three alternatives:
  - From the beginning to the first pipe character: `You can use the English`
  - From that pipe character to the next: `German`
  - From the last pipe character to the end: `Swedish interface`

To limit the boundaries of the alternatives you can enclose them in round brackets.

- Example: `Transit` interprets the regular expression `You can use the (English|German|Swedish) interface` as follows:
  - Normal text: `You can use the`
  - followed by the three alternatives: `English` or `German` or `Swedish`
  - followed by normal text: `interface`

Regular expression	Matches	Does not match
<code>(TermStar Transit) from STAR</code>	<ul style="list-style-type: none"> <li>● <code>TermStar</code> from <code>STAR</code></li> <li>● <code>Transit</code> from <code>STAR</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>TermStarTransit</code> from <code>STAR</code></li> <li>● <code>TermStar</code></li> </ul>
<code>TermStar Transit from STAR</code>	<ul style="list-style-type: none"> <li>● <code>TermStar</code></li> <li>● <code>Transit</code> from <code>STAR</code></li> </ul>	<ul style="list-style-type: none"> <li>● <code>TermStarTransit</code> from <code>STAR</code></li> <li>● <code>TermStar</code> from <code>STAR</code></li> </ul>

Search for alternatives

Alternatives and character groups/classes

Alternatives are not always useful or necessary in the following cases:

- Alternatives between individual characters
 

Alternatives between individual characters are not useful because the characters can be defined by a character group or class as part of which they will in any case be treated as alternatives.

  - Example: (a|b|c) means either the character a or b or c. Instead of that you can search for those alternatives by defining the character group [abc]
- Alternatives between character classes
 

Alternatives between character groups or classes are not useful because the characters can be defined by a character class as part of which they will in any case be treated as alternatives.

  - Example: ([a-z] | [0-9]) means any character from a to z or any character from 0 to 9. Instead of that you can search for those alternatives by defining the single character group [a-z0-9]

Transit does not interpret the pipe character as a meta character for alternatives when it is placed inside a character group or class.

When you use it inside a character group, Transit interprets the pipe character literally, i.e. it searches for the pipe character itself:

Regular expression	Matches	Does not match
st[ab x]r	<ul style="list-style-type: none"> <li>● star</li> <li>● stbr</li> <li>● st r</li> <li>● stxr</li> </ul>	<ul style="list-style-type: none"> <li>● str</li> <li>● stabr</li> <li>● staxr</li> </ul>

If the pipe character is inside a character group, it is interpreted literally

Alternatives and negated character strings

Alternatives between negated character strings are not generally useful. Such an approach would appear to make sense initially if want to exclude certain character strings from your search – however, it produces incorrect results.

Example:

You want to search for character strings that start with **A** followed by two-digit number. However, you do not want Transit to find character strings that contain 05 or 29.

You want Transit to find: A06, A09, A68

You do not want Transit to find: x7, A05, A29

- You therefore decide to use the following regular expression:

A(!05|!29)[0-9][0-9]

This regular expression, however, does not produce the desired results. It will also find A05 and A29.

Why? On the basis of that regular expression, Transit searches for the following character string:

- The letter **A**
- followed by a character sequence that is not 05 or is not 29
- followed by any number [0-9]
- followed again by any number [0-9]

At least one of the alternatives will always be matched.

- If the **A** is followed by the number 05, then that is a character string that is not 29. The second alternative is thus a match.
- If the **A** is followed by the number 29, then that is a character string that is not 05. The first alternative is thus a match.
- If the **A** is followed by another number, then that is a character string that is neither 29 nor 05. Both alternatives are thus a match.

This regular expression therefore does not prevent Transit finding A05 or A29.

- The desired result is obtained by using the following regular expression:

**A**(!05) (!29) [0-9] [0-9]

On the basis of that regular expression, Transit searches for the following character string:

- The letter **A**
- not followed by 05 and
- not followed by 29
- followed by any number [0-9]
- followed again by any number [0-9]

Now Transit proceeds as follows:

- If the **A** is followed by the number 05, the first negation means it is not a match because it specifies that **A** cannot be followed by 05. Transit thus does not find A05.
- If the **A** is followed by the number 29, the first negation allows a match (not followed by 05). However, the second negation prevents a match because it specifies that **A** cannot be followed by 29. Transit thus does not find A29.
- If the **A** is followed by another number, the first and second negations allow a match. Transit thus does find A06, A09, A68, etc.

## Variables: #

**Why use variables?** When performing a normal Find and Replace without using variables, you can only specify one string with which to replace the search string.

You can use variables in a Find and Replace operation to define variable components of the character string that Transit is to search for. That allows you to perform complex Find and Replace sequences in a single operation.

- Example: You want to replace the phrase `Year 2015` or `Year 2016` or `Year 2017` by `2015 Edition` or `2016 Edition` or `2017 Edition` respectively.

With a normal Find and Replace you would have to run three separate searches to replace `Year 2015` with `2015 Edition`, `Year 2016` with `2016 Edition` and `Year 2017` with `2017 Edition`.

If, however, you use the regular expression `Year 201#[([5-7])0]`, Transit will find any of the three phrases in the same search and substitute the appropriate replacement specified by the regular expression `201#0 Edition`. The precise meaning of the expressions is explained later on ([» How are variables used?](#), page 133).

Other practical examples:

- You want every single-digit number to have a leading zero:  
1 is to be changed to 01; 2 is to be changed to 02, etc.
- You want Transit to change all numbers in the format `x.y` to the format `x,y`:  
2.1 is to be changed to 2,1; 2.2 is to be changed to 2,2, etc.
- You want Transit to change all hyphenated combinations (`abc-xyz`) to the format `abc_xyz`.  
`0n-line` is to be changed to `0n_line`; `reference-based` changed to `reference_based`, etc.

For this, you have to define variables for such cases ([» How are variables used?](#), page 133).

When performing a Find and Replace using variables, you can also specify whether Transit is to convert the string when replacing:

- Changing the case when replacing ([» page 134](#))
- Changing number formats when replacing ([» page 135](#))
- Performing mathematical calculations when replacing ([» page 136](#))
- Rounding figures when replacing ([» page 138](#))
- Converting numbers to characters when replacing, and vice versa ([» page 139](#))

How are variables used? In order that it can use parts of the character string that it finds to replace what it finds, Transit has to memorise those items during the search.

To enable it to do so, you use variables which you define in the regular expression for the search and in which Transit stores components of the character string.

You then use the same variables in the regular expression for the replacement so that Transit uses the stored data when performing the replacement.

- Syntax for regular expression for search string:

`#(regular expression)variable number`

Transit will thus store the characters that it finds on the basis of the regular expression in the specified variable. You can use up to ten variables in a Find/Replace operation (variable numbers 0 to 9).

- Example: The following format is used to refer to keyboard keys in a document:  
Ins key, DEL key, PgUp key, return key, etc.

You want to run a Find and Replace so that the names of the keys are placed in quotation marks thus:

"Ins" key, "DEL" key, "PgUp" key, "return" key, etc.

You can use the following regular expression for the search:

`#([a-z]+)\skey`

On that basis Transit will search for a sequence of letters ([a-z]) followed by a space (\s) followed by the string key. Transit saves the sequence of letters found as the variable 0.

If you use only the variable number in the search and do not specify a wildcard character or character string, Transit uses the wildcard character & as the regular expression. That means that Transit will search for any sequence of any number of characters and will store it as the specified variable.

Since the wildcard character & always requires a beginning and end delimiter, you must specify them. However, they are not stored in the variable.

- Example: You use the expression `s#9r` for the search string

Transit interprets that expression as `s#(&)9r`. Transit therefore searches for `s` followed by any sequence of any number of characters followed by `r`. Transit stores that sequence of characters as the variable 9. The delimiters `s` and `r` are not stored in the variable.

- Syntax for regular expression for replacement string:

`#Variable number`

- Example: In the above example involving the format for the key names, you could use the following regular expression for the replacement:

`"#0" key`

Transit then replaces the string found with the character `"`, followed by the contents of the variable 0, followed by the string `" key`.

The following table lists further practical examples:

Application	Search	Replace
Assign a leading zero to single-digit numbers	<code>\s#([0-9])0</code> Transit searches for a space <code>\s</code> followed by a number <code>[0-9]</code> . Transit stores the number as the variable 0.	<code>\s0#0</code> Transit replaces with a space, a zero and the contents of the variable 0 (the number found).
Replace single-digit numbers in notation <code>x.y</code> by notation <code>x,y</code>	<code>\s#([0-9])0\.#([0-9])1</code> Transit searches for a space <code>\s</code> followed by a number <code>[0-9]</code> followed by a decimal point <code>\.</code> followed by a number <code>[0-9]</code> . Transit saves the first number as the variable 0 and the second number as the variable 1.	<code>\s#0,#1</code> Transit replaces with a space, the contents of the variable 0 (the first number found), a comma and the contents of the variable 1 (the second number found).
Replace the hyphen in compound words with an underscore	<code>#([a-z]+)0-#([a-z]+)1</code> Transit searches for a sequence of letters <code>[a-z]+</code> followed by a hyphen followed by a sequence of letters <code>[a-z]+</code> . Transit saves the first sequence of letters as the variable 0 and the second sequence of letters as the variable 1.	<code>#0_#1</code> Transit replaces with the contents of the variable 0 (the first sequence of letters found), an underscore and the contents of the variable 1 (the second sequence of letters found).

Replacements using variables

Changing the case when replacing

If you use variables, Transit can change the case of the variable contents when replacing.



**Upper and lower case**

To force Transit to distinguish between upper and lower case when searching, select „Match case“ option in Transit.

- Syntax for regular expression for replacement string:  
`#(<option>)variable number`  
In place of `<option>` you can enter one of the following options to specify how Transit is to change the case:

Option	Meaning
<code>^</code>	Change replacement string to all capitals
<code>^1</code>	Change first letter of replacement string to capital. Leave all other letters unchanged.
<code>_</code>	Change replacement string to all lower case

Options for changing case

Option	Meaning
_1	Change first letter of replacement string to lower case. Leave all other letters unchanged.
~	Invert case of all letters in replacement string (i.e. change lower case to upper case and upper case to lower case)
~1	Invert case of first letter in replacement string (i.e. change lower case to upper case or upper case to lower case). Leave all other letters unchanged.

Options for changing case (cont.)

- Example: The following format is used to refer to keyboard keys in a document: `Ins key`, `DEL key`, `PgUp key`, `return key`, etc. You want to use a Find and Replace to change the case of the key names.

To do so, you use the following regular expression for the search:

```
#([a-zA-Z]+)0\sk
```

The following table shows the results obtained by the various options for changing case.

Replace	Ins key	DEL key	PgUp key	return key
#(^)0 key	INS key	DEL key	PGUP key	RETURN key
#(^1)0 key	Ins key	DEL key	PgUp key	Return key
#(_)0 key	ins key	del key	pgup key	return key
#(_1)0 key	ins key	dEL key	pgUp key	rETURN key
#(-)0 key	INS key	del key	pGuP key	RETURN key
#(~1)0 key	ins key	dEL key	pgUp key	Return key

Examples of changing case

### Changing number formats when replacing

If you use variables, Transit can change the number format of the variable contents when replacing.

- Syntax for regular expression for replacement string:

```
#(={(<format>)}x)<variable number>
```

In place of `<format>` you enter a format option to specify how Transit is to change the number format.

Format	Meaning
#	Show number only if value is not zero
0	Always show number whether value is zero or not
_	Show space if value is zero

Format options for changing number format

If you do not specify a number format, Transit applies the number format specified in the Windows system settings.

- Example: A document contains decimal numbers in a variety of formats: 1.1, 10.123, etc. You want to use Find and Replace to change the format of the numbers.

To do so, you use the following regular expression for the search:

`#([0-9]+\.[0-9]+)0`

The following table shows the results obtained by the various options for changing number format.

Replace	1.1	10.123
<code>#(=[_ .- ]+\.00)x0</code>	1.10 (with four leading spaces)	10.12 (with three leading spaces)
<code>#(=[00000.00]x)0</code>	00001.10	00010.12
<code>#(=[0.##]x)0</code>	1.1	10.12
<code>#(=[}x)0</code>	1	10

Examples of changing number format



### Rounding numbers

If you reduce the number of decimal places, Transit cuts off the surplus decimal places. However, by specifying a calculation, you can have Transit round the number ([» Rounding figures when replacing](#), page 138).

### Performing mathematical calculations when replacing

If you use variables, Transit can perform mathematical calculations on the variable contents when replacing.

- Syntax for regular expression for replacement string:  
`#(<format><formula><variable number>`

You use `<format>` to specify how Transit is to change the number format, if required ([» Changing number formats when replacing](#), page 135). In our examples, however, we have left out the format option so as not to confuse the explanation of the mathematical calculations.

In place of `<formula>` you enter the mathematical formula that Transit is to apply to the number. In that formula you can use the following mathematical operators:

Operator	Meaning
+	Addition
-	Subtraction
/	Division
*	Multiplication
()	Brackets
x	Number to be converted

Operators for mathematical calculations

- Example: A document contains a variety of decimal numbers: 295200.2, 0.123, 3.4, etc. You want to use Find and Replace to perform a calculation on the numbers.



To do so, you use the following regular expression for the search:

#([0-9]+.[0-9]+)0

The following table shows the results obtained by various mathematical calculations.

Replace	295200,2	0,123	3,4
#(={})x)0	295200	0	3
Round off to whole number			
#(={})x+5)0	295205	5	8
Add 5			
#(={})x-5)0	295195	-4	-1
Subtract 5			
#(={})x*5)0	1476001	0	17
Multiply by 5			
#(={})x/5)0	59040	0	0
Divide by 5			
#(={})x*-9)0	-2632501	-1	-30
Multiply by -9 (negative value)			
#(={})x*x)0	87143153664	0	11
Multiply by itself (= x <sup>2</sup> )			

Examples of mathematical calculations



### Rounding numbers

If you reduce the number of decimal places, Transit cuts off the surplus decimal places. However, by specifying a calculation, you can have Transit round the number (» [Rounding figures when replacing](#), page 138).

Other practical examples:

- Replace dimensions in inch by dimensions in centimetres

A document contains dimensions in inch. They are to be replaced by the equivalent dimensions in centimetres.

- Regular expression for the search string:  
#[0-9]+0 inch
- Regular expression for the replacement string:  
#(={})x\*2.54)0 cm

Result:

The width is 5 inch.  
is replaced by  
The width is 12.7 cm.

- Where distance quoted in miles add equivalent distance in kilometres  
A document contains distances in miles. You want Transit to add the equivalent distance in kilometres in each case.

- Regular expression for the search string:  
`#([0-9]+)0 miles`
- Regular expression for the replacement string:  
`#0 miles (#(={})x*1.609)0 km)`

Result:

Distance Ramsen - Sindelfingen 90 miles  
is replaced by  
Distance Ramsen - Sindelfingen 90 miles (144 km)

**Rounding figures when replacing**

If you use variables, Transit can change the number format of the variable contents when replacing (» [Changing number formats when replacing](#), page 135).

If the change of format reduces the number of decimal places, Transit does not automatically round the number, it simply cuts off the surplus decimal places.

- Example: Reduce all decimal numbers to one decimal place  
A document contains decimal numbers with varying numbers of decimal places. You want to change them all to decimal numbers with only one decimal place.

- Regular expression for the search string:  
`#([0-9]+\.[0-9]+)0`
- Regular expression for the replacement string:  
`#(={0.0}x)0`

Result: Transit cuts off the surplus decimal places thus:

Number found	Reduced to one decimal place
1.01	1.0
1.04	1.0
1.05	1.0
1.06	1.0
1.09	1.0

Example: Reducing numbers to one decimal place

However, by specifying an additional calculation in the replacement, you can have Transit round the number: Values up to and including ...4 will be rounded down, values of ...5 or greater will be rounded up.

- Example: When replacing you add 0.05 to the numerical value. As a result Transit increases the numerical value by 0.05 when replacing and then cuts off the surplus decimal places. The resulting figures are thus rounded to the nearest tenth.
  - Regular expression for the search string (same as above):  
#([0-9]+\.[0-9]+)0
  - Regular expression for the replacement string:  
#(=[0.0]x+.05)0

Result: Transit rounds the values up or down to the nearest tenth

Number found	0.05 added	Reduced to one decimal place
1.01	1.06	1.0
1.04	1.09	1.0
1.05	1.10	1.1
1.06	1.11	1.1
1.09	1.14	1.1

Example: Rounding numbers to one decimal place

To determine the amount to be added, take the smallest decimal fraction required and divide it by 2.

- Example: Rounding to two decimal places: Smallest decimal fraction 0.01 divided by 2 equals 0.005.  
To round figures, add the value 0.005.

Converting numbers to characters when replacing, and vice versa

If you use variables, Transit can change numbers to characters or characters to numbers when replacing.

- Syntax for regular expression for replacement string:  
#(<option><variable number>

In place of <option>, you can enter one of the following values to specify how Transit should change numbers to characters, and vice versa.

Option	Meaning
c	Change a decimal number to character using the relevant Unicode
cx	Change a hexadecimal number to character using the relevant Unicode
n	Change character to the decimal Unicode for the character
nx	Change character to the hexadecimal Unicode for the character
nx2	Change character to a two-digit hexadecimal number
nx4	Change character to a four-digit hexadecimal number

Options for changing numbers to characters, and vice versa

## Invalid regular expressions

**Ambiguous regular expressions** Regular expressions must be unambiguous: There must only be one way in which they can be interpreted. If you use quantifiers the regular expression may be invalid because it may allow different interpretations so that the search does not produce a definite result.

Ambiguous expressions	Error	Examples for correct expressions
3*3	The second number 3 is already found by 3*	<ul style="list-style-type: none"> <li>● 33*</li> <li>● 3+</li> </ul>
[3-7]*4	4 is already found by [3-7]*	[3-7]
T.*ion	ion is already found by .*	T&ion

Examples of ambiguous regular expressions

**Syntax errors** The syntax of regular expressions can become very complex – particularly where the requirements of the search are themselves very complex. As a result, syntax errors can creep in and invalidate the regular expression.

Wrong syntax	Error	Examples for correct syntax
Wy\	The backslash is the last character. The backslash is a meta character (escapement) or introduces a control character. In either case, it has to be followed by another character (» <a href="#">Control characters</a> , page 112 and » <a href="#">Escapement: \</a> , page 121).	Wy\\
[9-0]+	The character class contains an invalid range. When you define a character class, you have to keep to the order in which the characters appear in the ANSI character table (» <a href="#">Wildcard for any of a specified group or class: [ ]</a> , page 114).	<ul style="list-style-type: none"> <li>● [0-9]+</li> <li>● [09]+</li> </ul>
[0-9]&	The ampersand is the last character. The ampersand is a meta character (wildcard character representing any sequence of any number of characters). It has to be bounded by beginning and end delimiters (» <a href="#">Wildcard for any sequence of characters: &amp;</a> , page 116).	<ul style="list-style-type: none"> <li>● [0-9]&amp;\$</li> <li>● [0-9].*\$</li> </ul>
key no	The pipe character is the last character. The pipe character is a meta character (alternative). It must be placed <u>between</u> two alternatives, i.e. it must be followed by the second alternative (» <a href="#">Alternatives:  </a> , page 129).	<ul style="list-style-type: none"> <li>● key no</li> <li>● key no yes</li> </ul>

Examples of typical syntax errors

Wrong syntax	Error	Examples for correct syntax
(key no	The closing bracket is missing. When you use brackets to structure regular expressions, every opening bracket must have a corresponding closing bracket (» <a href="#">Applying meta characters to character strings: ( )</a> , page 123).	(key no)
(key no]	Different types of bracket are mixed. Regular expressions permit the use of round brackets (for structuring) and square brackets (for character groups/classes). Make sure that you do not mix the two types (» <a href="#">Wildcard for any of a specified group or class: [ ]</a> , page 114 and » <a href="#">Applying meta characters to character strings: ( )</a> , page 123).	<ul style="list-style-type: none"> <li>● (key no)</li> <li>● (key  [no])</li> </ul>
*a	The asterisk is the first character. The asterisk is a meta character (quantifier) and specifies how many instances of the preceding character are to be found. Therefore, it has to be preceded by another character (» <a href="#">Quantifiers: + * ?</a> , page 118).	a*

Examples of typical syntax errors

# 14 Supported working languages

Transit/TermStar supports more than 200 working languages:

- Sorted by language name ([» page 142](#))
- Sorted by language code ([» page 148](#))

Sorted by  
language name

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
AFK	Afrikaans	0x0436	1078	af
SQI	Albanian	0x041C	1052	sq-al
AMH	Amharic	0x045E	1118	am-et
ARG	Arabic (Algeria)	0x1401	5121	ar-dz
ARH	Arabic (Bahrain)	0x3C01	15361	ar-bh
ARE	Arabic (Egypt)	0x0C01	3073	ar-eg
ARI	Arabic (Iraq)	0x0801	2049	ar-iq
ARJ	Arabic (Jordan)	0x2C01	11265	ar-jo
ARK	Arabic (Kuwait)	0x3401	13313	ar-kw
ARB	Arabic (Lebanon)	0x3001	12289	ar-lb
ARL	Arabic (Libya)	0x1001	4097	ar-ly
ARM	Arabic (Morocco)	0x1801	6145	ar-ma
ARO	Arabic (Oman)	0x2001	8193	ar-om
ARQ	Arabic (Qatar)	0x4001	16385	ar-qa
ARA	Arabic (Saudi Arabia)	0x0401	1025	ar-sa
ARS	Arabic (Syria)	0x2801	10241	ar-sy
ART	Arabic (Tunisia)	0x1C01	7169	ar-tn
ARU	Arabic (U.A.E.)	0x3801	14337	ar-ae
ARY	Arabic (Yemen)	0x2401	9217	ar-ye
HYE	Armenian	0x042B	1067	hy-am
ASM	Assamese	0x044D	1101	as-in
AZC	Azerbaijani (cyrillic)	0x082C	2092	az-Cyrl-az
AZE	Azerbaijani (latin)	0x042C	1068	az-Latn-az
EUQ	Basque	0x042D	1069	eu
BEL	Belarusian	0x0423	1059	be-by

Supported working languages: Sorted by language name

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
BNG	Bengali	0x0445	1093	bn-in
BOC	Bosnian (cyr., Bosn. and Herzeg.)	0x201A	8218	bs-Cyrl-ba
BOS	Bosnian (lat., Bosn. and Herzeg.)	0x141A	5146	bs-Latn-ba
BRE	Breton	0x047E	1150	br-fr
BGR	Bulgarian	0x0402	1026	bg-bg
MYA	Burmese	0x0455	1109	my-mm
CAT	Catalan	0x0403	1027	ca
KUR	Central Kurdish / Sorani	0x0492	1170	ku-iq
ZHH	Chinese (Hong Kong)	0x0C04	3076	zh-hk
ZHM	Chinese (Macau)	0x1404	5124	zh-mo
CHS	Chinese (PR China)	0x0804	2052	zh-cn
ZHI	Chinese (Singapore)	0x1004	4100	zh-sg
CHT	Chinese (Taiwan)	0x0404	1028	zh-tw
COS	Corsican	0x0483	1155	co-fr
HRV	Croatian	0x041A	1050	hr-hr
HRB	Croatian (Bosn. and Herzeg.)	0x101A	4122	hr-ba
CSY	Czech	0x0405	1029	cs-cz
DAN	Danish	0x0406	1030	da-dk
PRS	Dari	0x048C	1164	prs-af
NLD	Dutch	0x0413	1043	nl-nl
NLB	Dutch (Belgium)	0x0813	2067	nl-be
NLS	Dutch (special)	0x7C13	31763	nl
EDO	Edo	0x0466	1126	bin-ng
EFI	Efik/Ibibio	0x0469	1129	efi
ENA	English (Australia)	0x0C09	3081	en-au
ENL	English (Belize)	0x2809	10249	en-bz
ENC	English (Canada)	0x1009	4105	en-ca
ENB	English (Caribbean)	0x2409	9225	en-xx
ENH	English (Hong Kong)	0x3C09	15369	en-hk
END	English (India)	0x4009	16393	en-in
ENN	English (Indonesia)	0x3809	14345	en-id
ENI	English (Ireland)	0x1809	6153	en-ie
ENJ	English (Jamaica)	0x2009	8201	en-jm
ENM	English (Malaysia)	0x4409	17417	en-my
ENZ	English (New Zealand)	0x1409	5129	en-nz
ENP	English (Philippines)	0x3409	13321	en-ph
EN1	English (simplified)	0x7C09	31753	e1-gb

Supported working languages: Sorted by language name (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
ENO	English (Singapore)	0x4C09	19465	en-sg
ENS	English (South Africa)	0x1C09	7177	en-za
ENT	English (Trinidad and Tobago)	0x2C09	11273	en-tt
ENG	English (UK)	0x0809	2057	en-gb
ENU	English (US)	0x0409	1033	en-us
ENW	English (Zimbabwe)	0x3009	12297	en-zw
ETI	Estonian	0x0425	1061	et-ee
FOS	Faroese	0x0438	1080	fo
FIL	Filipino (Philippines)	0x0464	1124	fil-ph
FIN	Finnish	0x040B	1035	fi-fi
FRA	French	0x040C	1036	fr-fr
FRB	French (Belgium)	0x080C	2060	fr-be
FRO	French (Cameroon)	0x2C0C	11276	fr-cm
FRC	French (Canada)	0x0C0C	3084	fr-ca
FRG	French (Congo)	0x240C	9228	fr-cg
FRV	French (Cote d'Ivoire)	0x300C	12300	fr-ci
FRH	French (Haiti)	0x3C0C	15372	fr-ht
FRL	French (Luxembourg)	0x140C	5132	fr-lu
FRI	French (Mali)	0x340C	13324	fr-ml
FRM	French (Monaco)	0x180C	6156	fr-mc
FRR	French (Morocco)	0x380C	14348	fr-ma
FR1	French (rationalised)	0x7C0C	31756	f1-fr
FRU	French (Reunion)	0x200C	8204	fr-re
FRE	French (Senegal)	0x280C	10252	fr-sn
FRS	French (Switzerland)	0x100C	4108	fr-ch
FRW	French (West Indies)	0x1C0C	7180	fr-xx
FRY	Frisian (Netherlands)	0x0462	1122	fy-nl
FUB	Fulfulde/Adamawa	0x0467	1127	fub-cm
GAE	Gaelic (Ireland)	0x083C	2108	ga-ie
GDH	Gaelic (Scotland)	0x043C	1084	gd-gb
GAL	Gallegan/Galician	0x0456	1110	gl-es
KAT	Georgian	0x0437	1079	ka-ge
DEU	German	0x0407	1031	de-de
DEA	German (Austria)	0x0C07	3079	de-at
DEC	German (Liechtenstein)	0x1407	5127	de-li
DEL	German (Luxembourg)	0x1007	4103	de-lu
DE1	German (plain language)	0x7C07	31751	d1-de

Supported working languages: Sorted by language name (cont.)



Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
DES	German (Switzerland)	0x0807	2055	de-ch
ELL	Greek	0x0408	1032	el-gr
GRC	Greek (ancient)	0x7C08	31752	grc-gr
GUA	Guarani	0x0474	1140	gn-py
GUJ	Gujarati	0x0447	1095	gu-in
HAU	Hausa	0x0468	1128	ha-ng
HAW	Hawaiian	0x0475	1141	haw
HEB	Hebrew	0x040D	1037	he-il
HIN	Hindi	0x0439	1081	hi-in
HUN	Hungarian	0x040E	1038	hu-hu
ISL	Icelandic	0x040F	1039	is-is
IBO	Igbo	0x0470	1136	ig-ng
IND	Indonesian	0x0421	1057	id-id
ITA	Italian	0x0410	1040	it-it
ITS	Italian (Switzerland)	0x0810	2064	it-ch
JPN	Japanese	0x0411	1041	ja-jp
KAN	Kannada	0x044B	1099	kn-in
KAZ	Kazakh	0x043F	1087	kk-kz
KHM	Khmer	0x0453	1107	kh-kh
KIR	Kirghiz	0x0440	1088	ky-kg
KOR	Korean	0x0412	1042	ko-kr
LAO	Lao	0x0454	1108	lo-la
LAT	Latin	0x0476	1142	la
LVI	Latvian	0x0426	1062	lv-lv
LTH	Lithuanian	0x0427	1063	lt-lt
LBX	Luxembourgish	0x046E	1134	lb-lu
MKD	Macedonian (North Macedonia)	0x042F	1071	mk-mk
MSL	Malay	0x043E	1086	ms-my
MSB	Malay (Brunei Darussalam)	0x083E	2110	ms-bn
MAL	Malayalam	0x044C	1100	ml-in
MTL	Maltese	0x043A	1082	mt-mt
MRI	Maori	0x0481	1153	mi-nz
MAR	Marathi	0x044E	1102	mr-in
MNG	Mongolian	0x0450	1104	mn-mn
NDE	Ndebele (Northern)	0x08EE	2286	nd-zw
NBL	Ndebele (Southern)	0x04EE	1262	nr-za
KMR	Northern Kurdish / Kurmanji	0x04EB	1259	kmr

Supported working languages: Sorted by language name (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
NOR	Norwegian (Bokmal)	0x0414	1044	nb-no
NON	Norwegian (Nynorsk)	0x0814	2068	nn-no
OCI	Occitan	0x0482	1154	oc-fr
ORI	Oriya	0x0448	1096	or-in
ORO	Oromo	0x0472	1138	or-et
PAS	Pashto	0x0463	1123	ps-af
FAR	Persian	0x0429	1065	fa-ir
PLK	Polish	0x0415	1045	pl-pl
PTG	Portuguese	0x0816	2070	pt-pt
PTB	Portuguese (Brazil)	0x0416	1046	pt-br
PAN	Punjabi	0x0446	1094	pa-in
QUE	Quechua	0x04EF	1263	qu-py
ROM	Romanian	0x0418	1048	ro-ro
ROV	Romanian (Moldova)	0x0818	2072	ro-md
RMS	Romansh	0x0417	1047	rm-ch
RUS	Russian	0x0419	1049	ru-ru
RUM	Russian (Moldova)	0x0819	2073	ru-md
SZI	Sami	0x043B	1083	se
SAN	Sanskrit	0x044F	1103	sa-in
SRC	Serbian (cyr., Bosn. and Herzeg.)	0x1C1A	7194	sr-Cyrl-ba
SCM	Serbian (cyr., Montenegro)	0x301A	12314	sr-Cyrl-me
SRB	Serbian (cyrillic)	0x0C1A	3098	sr-Cyrl-rs
SRH	Serbian (lat., Bosn. and Herzeg.)	0x181A	6170	sr-Latn-ba
SRM	Serbian (lat., Montenegro)	0x2C1A	11290	sr-Latn-me
SRL	Serbian (latin)	0x081A	2074	sr-Latn-rs
SIN	Sinhala	0x045B	1115	si-lk
SSW	Siswati	0x04ED	1261	ss-za
SKY	Slovak	0x041B	1051	sk-sk
SLV	Slovenian	0x0424	1060	sl-si
SML	Somali	0x0477	1143	so-so
NSO	Sotho (Northern)	0x0830	2096	ns-za
SXT	Sotho (Southern)	0x0430	1072	st-za
ESP	Spanish	0x040A	1034	es-es
ESS	Spanish (Argentina)	0x2C0A	11274	es-ar
ESB	Spanish (Bolivia)	0x400A	16394	es-bo
ESL	Spanish (Chile)	0x340A	13322	es-cl
ESO	Spanish (Colombia)	0x240A	9226	es-co

Supported working languages: Sorted by language name (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
ESC	Spanish (Costa Rica)	0x140A	5130	es-cr
ESK	Spanish (Cuba)	0x5C0A	23562	es-cu
ESD	Spanish (Dominican Rep.)	0x1C0A	7178	es-do
ESF	Spanish (Ecuador)	0x300A	12298	es-ec
ESE	Spanish (El Salvador)	0x440A	17418	es-sv
ESG	Spanish (Guatemala)	0x100A	4106	es-gt
ESH	Spanish (Honduras)	0x480A	18442	es-hn
ES1	Spanish (International)	0x7C0A	31754	es-zz
ESM	Spanish (Mexico)	0x080A	2058	es-mx
ESI	Spanish (Nicaragua)	0x4C0A	19466	es-ni
ESA	Spanish (Panama)	0x180A	6154	es-pa
ESZ	Spanish (Paraguay)	0x3C0A	15370	es-py
ESR	Spanish (Peru)	0x280A	10250	es-pe
ESU	Spanish (Puerto Rico)	0x500A	20490	es-pr
ESY	Spanish (Uruguay)	0x380A	14346	es-uy
EST	Spanish (US)	0x540A	21514	es-us
ESV	Spanish (Venezuela)	0x200A	8202	es-ve
SWK	Swahili	0x0441	1089	sw-ke
SVE	Swedish	0x041D	1053	sv-se
TGL	Tagalog (Philippines)	0x04EC	1260	tl-ph
TAJ	Tajik	0x0428	1064	tg-tj
TAM	Tamil	0x0449	1097	ta-in
TEL	Telugu	0x044A	1098	te-in
THA	Thai	0x041E	1054	th-th
TGE	Tigrinya (Eritrea)	0x0873	2163	ti-er
TGY	Tigrinya (Ethiopia)	0x0473	1139	ti-et
TSG	Tsonga	0x0431	1073	ts-za
TNA	Tswana	0x0432	1074	tn-bw
TRK	Turkish	0x041F	1055	tr-tr
TKM	Turkmen	0x0442	1090	tk-tm
UKR	Ukrainian	0x0422	1058	uk-ua
URI	Urdu (India)	0x0820	2080	ur-in
URD	Urdu (Pakistan)	0x0420	1056	ur-pk
UZC	Uzbek (cyrillic)	0x0843	2115	uz-Cyrl-uz
UZB	Uzbek (latin)	0x0443	1091	uz-Latn-uz
VEN	Venda	0x0433	1075	ve-za
VIT	Vietnamese	0x042A	1066	vi-vn

Supported working languages: Sorted by language name (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
WEL	Welsh	0x0452	1106	cy-gb
XHS	Xhosa	0x0434	1076	xh-za
JII	Yiddish	0x043D	1085	yi-il
YBA	Yoruba	0x046A	1130	yo-ng
ZUL	Zulu	0x0435	1077	zu-za

Supported working languages: Sorted by language name (cont.)

Sorted by  
language code

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
AFK	Afrikaans	0x0436	1078	af
AMH	Amharic	0x045E	1118	am-et
ARA	Arabic (Saudi Arabia)	0x0401	1025	ar-sa
ARB	Arabic (Lebanon)	0x3001	12289	ar-lb
ARE	Arabic (Egypt)	0x0C01	3073	ar-eg
ARG	Arabic (Algeria)	0x1401	5121	ar-dz
ARH	Arabic (Bahrain)	0x3C01	15361	ar-bh
ARI	Arabic (Iraq)	0x0801	2049	ar-iq
ARJ	Arabic (Jordan)	0x2C01	11265	ar-jo
ARK	Arabic (Kuwait)	0x3401	13313	ar-kw
ARL	Arabic (Libya)	0x1001	4097	ar-ly
ARM	Arabic (Morocco)	0x1801	6145	ar-ma
ARO	Arabic (Oman)	0x2001	8193	ar-om
ARQ	Arabic (Qatar)	0x4001	16385	ar-qa
ARS	Arabic (Syria)	0x2801	10241	ar-sy
ART	Arabic (Tunisia)	0x1C01	7169	ar-tn
ARU	Arabic (U.A.E.)	0x3801	14337	ar-ae
ARY	Arabic (Yemen)	0x2401	9217	ar-ye
ASM	Assamese	0x044D	1101	as-in
AZC	Azerbaijani (cyrillic)	0x082C	2092	az-Cyrl-az
AZE	Azerbaijani (latin)	0x042C	1068	az-Latn-az
BEL	Belarusian	0x0423	1059	be-by
BGR	Bulgarian	0x0402	1026	bg-bg
BNG	Bengali	0x0445	1093	bn-in
BOC	Bosnian (cyr., Bosn. and Herzeg.)	0x201A	8218	bs-Cyrl-ba
BOS	Bosnian (lat., Bosn. and Herzeg.)	0x141A	5146	bs-Latn-ba
BRE	Breton	0x047E	1150	br-fr

Supported working languages: Sorted by language code

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
CAT	Catalan	0x0403	1027	ca
CHS	Chinese (PR China)	0x0804	2052	zh-cn
CHT	Chinese (Taiwan)	0x0404	1028	zh-tw
COS	Corsican	0x0483	1155	co-fr
CSY	Czech	0x0405	1029	cs-cz
DAN	Danish	0x0406	1030	da-dk
DE1	German (plain language)	0x7C07	31751	d1-de
DEA	German (Austria)	0x0C07	3079	de-at
DEC	German (Liechtenstein)	0x1407	5127	de-li
DEL	German (Luxembourg)	0x1007	4103	de-lu
DES	German (Switzerland)	0x0807	2055	de-ch
DEU	German	0x0407	1031	de-de
EDO	Edo	0x0466	1126	bin-ng
EFI	Efik/Ibibio	0x0469	1129	efi
ELL	Greek	0x0408	1032	el-gr
EN1	English (simplified)	0x7C09	31753	e1-gb
ENA	English (Australia)	0x0C09	3081	en-au
ENB	English (Caribbean)	0x2409	9225	en-xx
ENC	English (Canada)	0x1009	4105	en-ca
END	English (India)	0x4009	16393	en-in
ENG	English (UK)	0x0809	2057	en-gb
ENH	English (Hong Kong)	0x3C09	15369	en-hk
ENI	English (Ireland)	0x1809	6153	en-ie
ENJ	English (Jamaica)	0x2009	8201	en-jm
ENL	English (Belize)	0x2809	10249	en-bz
ENM	English (Malaysia)	0x4409	17417	en-my
ENN	English (Indonesia)	0x3809	14345	en-id
ENO	English (Singapore)	0x4C09	19465	en-sg
ENP	English (Philippines)	0x3409	13321	en-ph
ENS	English (South Africa)	0x1C09	7177	en-za
ENT	English (Trinidad and Tobago)	0x2C09	11273	en-tt
ENU	English (US)	0x0409	1033	en-us
ENW	English (Zimbabwe)	0x3009	12297	en-zw
ENZ	English (New Zealand)	0x1409	5129	en-nz
ES1	Spanish (International)	0x7C0A	31754	es-zz
ESA	Spanish (Panama)	0x180A	6154	es-pa
ESB	Spanish (Bolivia)	0x400A	16394	es-bo

Supported working languages: Sorted by language code (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
ESC	Spanish (Costa Rica)	0x140A	5130	es-cr
ESD	Spanish (Dominican Rep.)	0x1C0A	7178	es-do
ESE	Spanish (El Salvador)	0x440A	17418	es-sv
ESF	Spanish (Ecuador)	0x300A	12298	es-ec
ESG	Spanish (Guatemala)	0x100A	4106	es-gt
ESH	Spanish (Honduras)	0x480A	18442	es-hn
ESI	Spanish (Nicaragua)	0x4C0A	19466	es-ni
ESK	Spanish (Cuba)	0x5C0A	23562	es-cu
ESL	Spanish (Chile)	0x340A	13322	es-cl
ESM	Spanish (Mexico)	0x080A	2058	es-mx
ESO	Spanish (Colombia)	0x240A	9226	es-co
ESP	Spanish	0x040A	1034	es-es
ESR	Spanish (Peru)	0x280A	10250	es-pe
ESS	Spanish (Argentina)	0x2C0A	11274	es-ar
EST	Spanish (US)	0x540A	21514	es-us
ESU	Spanish (Puerto Rico)	0x500A	20490	es-pr
ESV	Spanish (Venezuela)	0x200A	8202	es-ve
ESY	Spanish (Uruguay)	0x380A	14346	es-uy
ESZ	Spanish (Paraguay)	0x3C0A	15370	es-py
ETI	Estonian	0x0425	1061	et-ee
EUQ	Basque	0x042D	1069	eu
FAR	Persian	0x0429	1065	fa-ir
FIL	Filipino (Philippines)	0x0464	1124	fil-ph
FIN	Finnish	0x040B	1035	fi-fi
FOS	Faroese	0x0438	1080	fo
FR1	French (rationalised)	0x7C0C	31756	f1-fr
FRA	French	0x040C	1036	fr-fr
FRB	French (Belgium)	0x080C	2060	fr-be
FRC	French (Canada)	0x0C0C	3084	fr-ca
FRE	French (Senegal)	0x280C	10252	fr-sn
FRG	French (Congo)	0x240C	9228	fr-cg
FRH	French (Haiti)	0x3C0C	15372	fr-ht
FRI	French (Mali)	0x340C	13324	fr-ml
FRL	French (Luxembourg)	0x140C	5132	fr-lu
FRM	French (Monaco)	0x180C	6156	fr-mc
FRO	French (Cameroon)	0x2C0C	11276	fr-cm
FRR	French (Morocco)	0x380C	14348	fr-ma

Supported working languages: Sorted by language code (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
FRS	French (Switzerland)	0x100C	4108	fr-ch
FRU	French (Reunion)	0x200C	8204	fr-re
FRV	French (Cote d'Ivoire)	0x300C	12300	fr-ci
FRW	French (West Indies)	0x1C0C	7180	fr-xx
FRY	Frisian (Netherlands)	0x0462	1122	fy-nl
FUB	Fulfulde/Adamawa	0x0467	1127	fub-cm
GAE	Gaelic (Ireland)	0x083C	2108	ga-ie
GAL	Gallegan/Galician	0x0456	1110	gl-es
GDH	Gaelic (Scotland)	0x043C	1084	gd-gb
GRC	Greek (ancient)	0x7C08	31752	grc-gr
GUA	Guarani	0x0474	1140	gn-py
GUJ	Gujarati	0x0447	1095	gu-in
HAU	Hausa	0x0468	1128	ha-ng
HAW	Hawaiian	0x0475	1141	haw
HEB	Hebrew	0x040D	1037	he-il
HIN	Hindi	0x0439	1081	hi-in
HRB	Croatian (Bosn. and Herzeg.)	0x101A	4122	hr-ba
HRV	Croatian	0x041A	1050	hr-hr
HUN	Hungarian	0x040E	1038	hu-hu
HYE	Armenian	0x042B	1067	hy-am
IBO	Igbo	0x0470	1136	ig-ng
IND	Indonesian	0x0421	1057	id-id
ISL	Icelandic	0x040F	1039	is-is
ITA	Italian	0x0410	1040	it-it
ITS	Italian (Switzerland)	0x0810	2064	it-ch
JII	Yiddish	0x043D	1085	yi-il
JPN	Japanese	0x0411	1041	ja-jp
KAN	Kannada	0x044B	1099	kn-in
KAT	Georgian	0x0437	1079	ka-ge
KAZ	Kazakh	0x043F	1087	kk-kz
KHM	Khmer	0x0453	1107	kh-kh
KIR	Kirghiz	0x0440	1088	ky-kg
KMR	Northern Kurdish / Kurmanji	0x04EB	1259	kmr
KOR	Korean	0x0412	1042	ko-kr
KUR	Central Kurdish / Sorani	0x0492	1170	ku-iq
LAO	Lao	0x0454	1108	lo-la
LAT	Latin	0x0476	1142	la

Supported working languages: Sorted by language code (cont.)

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
LBX	Luxembourgish	0x046E	1134	lb-lu
LTH	Lithuanian	0x0427	1063	lt-lt
LVI	Latvian	0x0426	1062	lv-lv
MAL	Malayalam	0x044C	1100	ml-in
MAR	Marathi	0x044E	1102	mr-in
MKD	Macedonian (North Macedonia)	0x042F	1071	mk-mk
MNG	Mongolian	0x0450	1104	mn-mn
MRI	Maori	0x0481	1153	mi-nz
MSB	Malay (Brunei Darussalam)	0x083E	2110	ms-bn
MSL	Malay	0x043E	1086	ms-my
MTL	Maltese	0x043A	1082	mt-mt
MYA	Burmese	0x0455	1109	my-mm
NBL	Ndebele (Southern)	0x04EE	1262	nr-za
NDE	Ndebele (Northern)	0x08EE	2286	nd-zw
NLB	Dutch (Belgium)	0x0813	2067	nl-be
NLD	Dutch	0x0413	1043	nl-nl
NLS	Dutch (special)	0x7C13	31763	nl
NON	Norwegian (Nynorsk)	0x0814	2068	nn-no
NOR	Norwegian (Bokmal)	0x0414	1044	nb-no
NSO	Sotho (Northern)	0x0830	2096	ns-za
OCI	Occitan	0x0482	1154	oc-fr
ORI	Oriya	0x0448	1096	or-in
ORO	Oromo	0x0472	1138	or-et
PAN	Punjabi	0x0446	1094	pa-in
PAS	Pashto	0x0463	1123	ps-af
PLK	Polish	0x0415	1045	pl-pl
PRS	Dari	0x048C	1164	prs-af
PTB	Portuguese (Brazil)	0x0416	1046	pt-br
PTG	Portuguese	0x0816	2070	pt-pt
QUE	Quechua	0x04EF	1263	qu-py
RMS	Romansh	0x0417	1047	rm-ch
ROM	Romanian	0x0418	1048	ro-ro
ROV	Romanian (Moldova)	0x0818	2072	ro-md
RUM	Russian (Moldova)	0x0819	2073	ru-md
RUS	Russian	0x0419	1049	ru-ru
SAN	Sanskrit	0x044F	1103	sa-in
SCM	Serbian (cyr., Montenegro)	0x301A	12314	sr-Cyrl-me

Supported working languages: Sorted by language code (cont.)



Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
SIN	Sinhala	0x045B	1115	si-lk
SKY	Slovak	0x041B	1051	sk-sk
SLV	Slovenian	0x0424	1060	sl-si
SML	Somali	0x0477	1143	so-so
SQI	Albanian	0x041C	1052	sq-al
SRB	Serbian (cyrillic)	0x0C1A	3098	sr-Cyrl-rs
SRC	Serbian (cyr., Bosn. and Herzeg.)	0x1C1A	7194	sr-Cyrl-ba
SRH	Serbian (lat., Bosn. and Herzeg.)	0x181A	6170	sr-Latn-ba
SRL	Serbian (latin)	0x081A	2074	sr-Latn-rs
SRM	Serbian (lat., Montenegro)	0x2C1A	11290	sr-Latn-me
SSW	Siswati	0x04ED	1261	ss-za
SVE	Swedish	0x041D	1053	sv-se
SWK	Swahili	0x0441	1089	sw-ke
SXT	Sotho (Southern)	0x0430	1072	st-za
SZI	Sami	0x043B	1083	se
TAJ	Tajik	0x0428	1064	tg-tj
TAM	Tamil	0x0449	1097	ta-in
TEL	Telugu	0x044A	1098	te-in
TGE	Tigrinya (Eritrea)	0x0873	2163	ti-er
TGL	Tagalog (Philippines)	0x04EC	1260	tl-ph
TGY	Tigrinya (Ethiopia)	0x0473	1139	ti-et
THA	Thai	0x041E	1054	th-th
TKM	Turkmen	0x0442	1090	tk-tm
TNA	Tswana	0x0432	1074	tn-bw
TRK	Turkish	0x041F	1055	tr-tr
TSG	Tsonga	0x0431	1073	ts-za
UKR	Ukrainian	0x0422	1058	uk-ua
URD	Urdu (Pakistan)	0x0420	1056	ur-pk
URI	Urdu (India)	0x0820	2080	ur-in
UZB	Uzbek (latin)	0x0443	1091	uz-Latn-uz
UZC	Uzbek (cyrillic)	0x0843	2115	uz-Cyrl-uz
VEN	Venda	0x0433	1075	ve-za
VIT	Vietnamese	0x042A	1066	vi-vn
WEL	Welsh	0x0452	1106	cy-gb
XHS	Xhosa	0x0434	1076	xh-za
YBA	Yoruba	0x046A	1130	yo-ng
ZHH	Chinese (Hong Kong)	0x0C04	3076	zh-hk

Supported working languages: Sorted by language code (cont.)

## 14 SUPPORTED WORKING LANGUAGES

---

Language code	Language	Microsoft Locale ID (LCID)		ISO 639/3166 Code
		Hexadecimal	Decimal	
ZHI	Chinese (Singapore)	0x1004	4100	zh-sg
ZHM	Chinese (Macau)	0x1404	5124	zh-mo
ZUL	Zulu	0x0435	1077	zu-za

Supported working languages: Sorted by language code (cont.)





[www.star-group.net](http://www.star-group.net)

STAR-Group – Your single-source communication partner for products and services