Troi Text Plug-in 3.0 for FileMaker Pro 7, 8 and 8.5 USER GUIDE

August 2006

Troi Automatisering
Vuurlaan 18
2408 NB Alphen a/d Rijn
The Netherlands
Telephone: +31-172-426 606 Fax: +31-172-470 539

You can also visit the Troi web site at: <http://www.troi.com/> for additional information.

Table of Contents

Installing plug-ins

For Mac OS X:

- Quit FileMaker Pro.
- Put the file "Troi_Text.fmplugin" from the folder "Mac OS Plug-in" into the "Extensions" folder in the FileMaker Pro application folder.
- If you have installed previous versions of this plug-in, you are asked: “An older item named “Troi_Text.fmplugin” already exists in this location. Do you want to replace it with the one you’re moving?’. Press the OK button.
- Start FileMaker Pro. The first time the Troi Text Plug-in is used it will display a flash dialog box, indicating that it is loading and showing the registration status.

For Windows:

- Quit FileMaker Pro.
- Put the file "Troi_Text_Plugin.fmx" from the directory "Windows Plug-in" into the "Extensions" subdirectory in the FileMaker Pro application directory.
- If you have installed previous versions of this plug-in, you are asked: “This folder already contains a file called 'Troi_Text_Plugin.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The Troi Text Plug-in will display a dialog box, indicating that it is loading and showing the registration status.

The instructions above show FileMaker 7.You can also install the plug-in with FileMaker Pro 8 or 8.5.

TIP You can check which plug-ins you have loaded by going to the plug-in preferences. Do one of the following:

- Windows: choose Edit menu > Preferences.
- Mac OS: choose FileMaker Pro menu > Preferences.

Then in the Preferences dialog box, click the Plug-Ins tab.

You can now open the file "All Text Examples.fp7" to see how to use the plug-in's functions. There is also a function overview available.

If you have problems

This user guide tries to give you all the information necessary to use this plug-in. So if you have a problem please read this user guide first. Also you might visit our support web page:

<http://www.troi.com/support/>

This page contains FAQ's (Frequently Asked Questions), help on registration and much more. If that doesn't help you can get free support by email. Send your questions to **support@troi.com** with a full explanation of the problem. Also give as much relevant information (version of the plug-in, which platform, version of the operating system, version of FileMaker Pro) as possible.

If you find any mistakes in this manual or have a suggestion please let us know. We appreciate your feedback!

TIP You can get more information on returned error codes from our OSErrrs database on our web site: <http://www.troi.com/software/oserrrs.html>. This free FileMaker database lists all error codes for Windows and Mac OS!

What can this plug-in do?

The Troi Text Plug-in is a very powerful tool for dealing efficiently with text in your FileMaker Pro database. All from within FileMaker you can:

- Create combinational sets from 2 text fields:
  - get all lines that are the same
  - get all lines that differ
  - and other combinations
- Get (unique) lines and sort words and lines
- Make a text sum from a related file that updates without a script
- Parse XML Text

and more...

Software requirements

System requirements for Mac OS

Mac OS X 10.3.9 for PowerPC-based Macs or Mac OS X 10.4.5 for Intel-based Macs.

System requirements for Windows

Software Requirements: Windows 2000 (Service Pack 4), or Windows XP (Service Pack 2).

FileMaker requirements

FileMaker Pro 7 or FileMaker Developer 7. or higher
FileMaker Pro 8 or FileMaker Pro Advanced 8 or higher
FileMaker Pro 8.5 or FileMaker Pro Advanced 8.5 or higher.

NOTE Troi Text Plug-in version 2.8 (and later) uses the plug-in API introduced with FileMaker Pro 7. The functions of this plug-in have this format: FunctionName(parameter1 ; parameter2). This means that Unicode is supported and more.

You can also use FileMaker Server to serve databases that use functions of the Troi Text Plug-in. You need to have the plug-in installed at the clients that use these functions.

Troi Text Plug-in version 2.8 (and higher) does **not** run on versions prior to FileMaker Pro 7.0. If you need to run on versions prior to FileMaker Pro 7: see our web site for the Text Plug-in 2.1.6 which is using the 'classic' plug-in API, which is using the External("functionName" , "parameter") format. The 2.1.6 version runs on FileMaker 6, 5.x and 4.x.

FileMaker Server and AutoUpdate

You can also use FileMaker Server 7 or 8 to serve databases that use functions of the Troi Text Plug-in. You need to have the plug-in installed at the clients that use these functions.

The AutoUpdate feature of FileMaker Server 7 or 8 can help you automate installing and updating plug-ins automatically. We created an example file and a tar formatted plug-in of Troi Text Plug-in (only needed on Mac OS) to get you started:

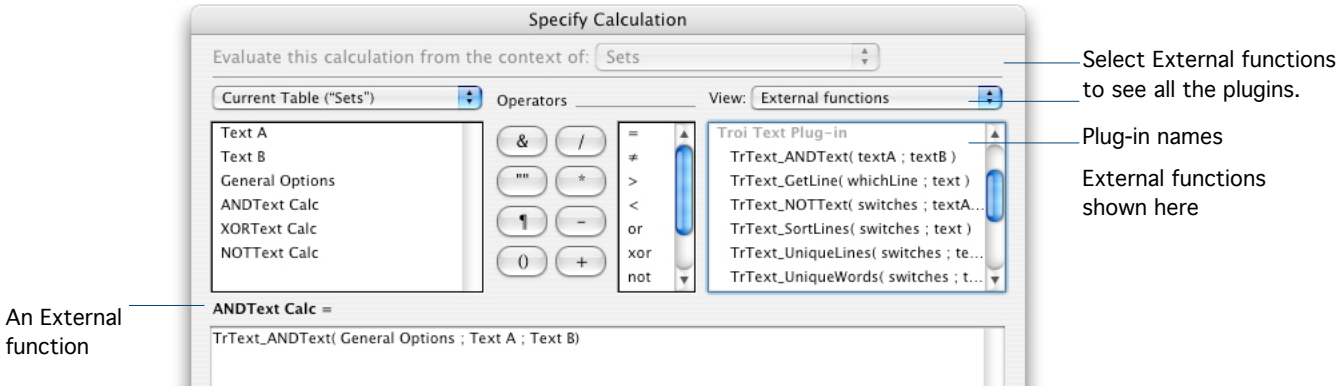
Visit our AutoUpdate web page to download the example:

<http://www.troi.com/software/autoupdate.html>

Getting started

Using external functions

The Troi Text Plug-in adds new functions to the standard functions that are available in FileMaker Pro. The functions added by a plug-in are called external functions. You can see those extra functions for all plug-ins at the top right of the Specify Calculation box:



You use special syntax with external functions: FunctionName(parameter1 ; parameter 2) where FunctionName is the name of an external function. A function can have zero or more parameters. Each parameter is separated by a semi-colon. Plug-ins don't work directly after installation. To access a plug-in function, you need to add the calls to the function in a calculation, for example in a text calculation in Define Fields or in a ScriptMaker Script.

Where to add the external functions?

External functions for this plug-in can be used in a calculation field when you are defining fields (choose Define Database from the File menu). Also the plug-in's functions can be used in a script step using a calculation, for example in a Set Field script step.

IMPORTANT The SumText functions have to be used in a specific way, to create the desired effect. See the section on SumText functions for the specifics on this.

Simple example

We start with a simple example to get you started. Say you have a database Pages.fp7, with a text field called myText. You want to list al unique words from myText. Go to Define Fields and define this calculation:

myUniqueWords Calculation = TrText_UniqueWords ("-Unused" ; myText)

Put the newly created calculation field on the layout.

if the contents of myText is "An ape is an ape, a rose is a rose" myUniqueWords will contain:

An
Ape
Is
A
Rose

Please take a close look at the included example files, as they provide a great starting point. From there you can move on, using the functions of the plug-in as building blocks. Together they give you powerful text tools.

Using SumText Functions

What SumText can do

The purpose of the three SumText functions is to get the concatenation of all the text of a text field in a related file. The 3 functions (TrText_SumTextStart, TrText_SumTextCalc and TrText_SumTextResult) work together to achieve this result. Before describing how to define SumText fields we give you two examples of what is possible:

Example 1

Say you have a file "Companies.fp7" holding company information and "Person.fp7" holding names and data of Persons. Companies.fp7 has a relation "Contacts" that relates all the contact persons of a company. So for company UFP you might have the following related contact persons:

ID	Name
1	Jean-Luc Picard
2	Lt. Commander Worf
3	Geordy LaForge

With the SumText functions it is possible to define a field that captures all the contact persons in one field. As a separator we use ", ". So in the file "Companies.fp7" you can define a sumtext field "All contacts" that has the following contents:

"Jean-Luc Picard, Lt. Commander Worf, Geordy LaForge"

And if someone creates a new contact person, say "Dr. Pulaski", the field is automatically (without running a script) updated to:

"Jean-Luc Picard, Lt. Commander Worf, Geordy LaForge, Dr. Pulaski"

The contents of this field will always reflect the current data of the related file.

Example 2

Using the same files and relations as in Example 1, but now in Contacts there is a third field called Favorites:

ID	Name	Favorites
1	Jean-Luc Picard	Earl Grey Tea
2	Lt. Commander Worf	NaPlah
3	Geordy LaForge	Twinkies
4	Dr. Pulaski	Ice Cream
5	Dr. Crusher	Earl Grey Tea

With the SumText functions it is possible to define a field that captures all the favorites of the contact persons in one field. This time the separator is a return "¶". In the file "Companies.fp3" you define a sumtext field "All Favorites" that has the following contents:

Earl Grey Tea ¶
NaPlah ¶
Twinkies ¶
Ice Cream ¶
Earl Grey Tea

In a script you can easily copy this list of favorites to a global "gFavorites", and then use this as a multikey to find all the related favorites.

Defining SumText Functions

NOTE With the the release of the update of FileMaker Pro 8.0v2 the original syntax of the SumText calculation is now again working properly. As the original syntax is more general we strongly recommend to use this syntax. Below is this syntax described.

These are the steps to make SumText work:

1- In your related file define a calculation field.

Go to “Define Fields” and create a new calculation field. Here this field is named “cSumTextCalc”. Use the "TrText_SumTextCalc" function in the calculation, like this:

```
cSumTextCalc = TrText_SumTextCalc( TextField )
```

Here “TextField” is the name of the field you want to sum. This new calculation will do the actual concatenation. Use a new field for each SumText calculation instance that will call from the main table.

IMPORTANT Make sure you make this field both unstored and a number, otherwise SumText won’t work!

2- In the main file define a field for the SumText result.

Use the following calculation:

```
sumText = Left( TrText_SumTextStart( "¶" ) &
          Sum(RelationName::cSumTextCalc) ; 0 ) &
          TrText_SumTextResult( "" )
```

In this calculation the function TrText_SumTextStart signals the plug-in to start summing. It takes a separator as parameter. The part Sum(RelationName::cSumTextCalc) makes FileMaker call all the related fields and finally TrText_SumTextResult signals to the plug-in that all has been summed and the result can be returned.

NOTE You no longer need to use a different sumTextInstanceID, please leave the sumTextInstanceID parameter empty. See the SumText.fp7 example file which uses three calculations.

You can also use this sumtext calculation in a SetField script step or any other calculation in the main file.

Why This Way?

You might say, why not make this simply one function like:

```
TrText_SumText(RelationName::TextField).
```

This would be easier, but this is not possible with the current plug-in implementation of FileMaker.

Summary of functions

The Troi Text Plug-in adds the following functions:

function name	short description
TrText_ANDText	Returns all lines that are both in text1 and text2.
TrText_GetLine	Returns the n-th line of the text.
TrText_NOTText	Returns all the lines in text1 that are NOT in text2.
TrText_SortLines	Returns all the lines from TextField sorted in ascending or descending alphabetical order.
TrText_SumTextCalc	Use this function to define a sumtextCalc help field in a related file.
TrText_SumTextResult	Use this function to stop the sumText calculation and get the result.
TrText_SumTextStart	Use this function to start the sumText calculation.
TrText_UniqueLines	Returns all unique lines from a text.
TrText_UniqueWords	Returns all unique words of a text.
TrText_Version	Use this function to see which version of the plug-in is loaded and to register the plug-in.
TrText_XML	Returns the requested parts from text formatted as XML.
TrText_XORText	Returns all the lines that are NOT both in text1 and text2.

Function Reference
TrText_ANDText

Syntax result = TrText_ANDText(switches ; text1 ; text2)

Returns all the lines that are both in text1 and text2.

Parameters

switches	(optional) determine way the result is returned
text1	first text
text2	second text

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

All the lines that are both in text1 and text2.

Special considerations

This is one of the 3 set manipulation functions.

Example usage

Set Field [result, TrText_ANDText("-Unused" ; Text1 ; Text2)]

If Text1 contains:
AA
BB
and Text2 contains:
BB
CC
then the result will be:
BB

Example 2

Text field "Text1" consists of the following lines:
1
10
12

And field "Text2" consists of the these lines:
1
2

The result of ANDText will be:
1

TrText_GetLine

Syntax result = TrText_GetLine(switches ; lineNumber ; theText)

Returns the n-th line of the text. The end of a line is determined by the return character ¶. This function returns all characters including the return character.

Parameters

switches	(optional) determine way the result is returned
lineNumber	the number of the line you want
theText	the text which you want to get a line from

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

The n-th line of text: all characters including the return character.

An empty line will result in a single return character.

If you request a line that is not in the text an empty text is returned. This makes it easy to use this function in a loop: start with line number 1 and increase until the result is empty.

Special considerations

The formatting in a layout does not alter the text field's contents. So if a field is formatted small, the lines may be wrapped of that layout but this does not enter extra returns into the text field.
The calculation in which this function is used can be both stored and unstored. You can use this function in function definitions or in ScriptMaker's Set Field Step.

Example usage

result = TrText_GetLine("-ReturnAtEnd" ; 2 ; "abc¶def¶ghi")

this will return as result the second line: "def¶"

Example 2

We assume that in your FileMaker file the following fields are defined:

TheText	Text
gRequestedLineNo	Global, number
gLine	Global, text

Then: TrText-GetLine(gRequestedLineNo ; TheText) will return the line indicated by gRequestedLineNo. This can be used in a ScriptMaker Script to extract single lines:

```
Set Field [gRequestedLineNo ; 1]
   Loop
    Set Field [gLine; TrText_GetLine( "-Unused" ; gRequestedLineNo ; TheText) ]
     Exit Loop If [ gLine = ""  ]
     Comment [ Do your stuff here... ]
    Set Field [gRequestedLineNo ; gRequestedLineNo + 1]
```

End Loop

Example 3

Text field "Text1" consists of the following lines:

line 1
this is line 2.
line 3 is line 3!

line5 (line 4 is empty).

The result of TrText_GetLine(gRequestedLineNo ; Text1) depends on the value of gRequestedLineNo:

gRequestedLineNo:	Returned result:	Remarks:
1	line 1¶	
2	this is line 2.¶	
3	line 3 is line 3!¶	
4	¶	only a return character is returned
5	line5 (line 4 is empty).¶	
6		no more lines: result is empty
0		invalid number: result is empty

TrText_NOTText

Syntax result = TrText_NOTText(switches; text1 ; text2)

Returns all the lines in text1 that are NOT in text2.

Parameters

switches	(optional) determine way the result is returned
text1	first text
text2	second text

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

All lines in text1 that are NOT in text2.

Special considerations

The order in which you give the 2 text parameters is important, unlike ANDText and XORText.
This is one of the 3 set manipulation functions.

Example usage

Set Field [result ; TrText_NOTText("-Unused" ; Text1 ; Text2)]

If Text1 contains:
AA
BB
and Text2 contains:
BB
CC
then the result will be:
AA

TrText_SortLines

Syntax result = TrText_SortLines(switches ; text)

Returns all lines from theText sorted in alphabetical order. The direction of the sort is determined by the switches.

Parameters

switches	determine the sort direction
theText	the text to be sorted

Switches can be one of:

-Ascending	(default) sort in ascending order (A...Z)
-Descending	sort in descending order (Z...A)

You can also add one of these switches:

-SortUnicodeRaw	sort with the raw unicode bytes (like the old ASCII ordering)
-SortUnicodeFMP	sort using FileMaker's way (SLOW)
-SortUnicodeTroi	(default) sort using Troi's sorting

You can also add this extra switch:

-ReturnAtEnd	add a return at the end

Returned result

All the lines from TextField sorted in alphabetical order.

Special considerations

If switches is empty the sort order is ascending.

The calculation in which this function is used can be both stored and unstored. You can use this function in calculation field definitions or in script calculations.

Example usage

TrText_SortLines("-Descending" ; Text)

Say text field "Text" consists of date strings (YYYY-MMDD), for example the following lines:
2000-1221
1999-0529
2000-1226

The result of SortLines will be:
2000-1226
2000-1221
1999-0529

In this case the latest date will be sorted first.

Example 2

We assume that in your FileMaker file the following fields are defined:

Text	Text
gDirection	Global, text

Create the following calculation:

```
Set Field [ result, TrText_SortLines( gDirection ; Text) ]
```

Put the field gDirection on the layout and create a valuelist with "-Descending". Format the field as a checkbox for it. Then by checking and unchecking gDirection you can instantly change the sort direction.

TrText_SumTextCalc

Syntax Set Field [result ; TrText_SumTextCalc(TextField)]

Use this function to define a sumtextCalc help field in a related file.

Parameters

The TextField is the field you want to create the text sum of.

Returned result

the result is coded, this function works together with the 2 other functions.

Special considerations

Please make sure you make it both unstored and a number, otherwise this won't work!

Example usage

cSumTextCalc = Calculation, Unstored, Number, = TrText_SumTextCalc(TextField).

See "Using sumText" for a detailed instruction on how to use sumText.

TrText_SumTextResult

Syntax Set Field [result ; TrText_SumTextResult(unused)]

Use this function to stop the sumText calculation and get the result.

Parameters

unusedplease leave empty. Use the calculation as specified in "Using SumText".

Returned result

The concatenated result of the SumText function.

Example usage

See "Using SumText" for an overview of how to use SumText.

TrText_SumTextStart

Syntax Set Field [result; TrText_SumTextStart(separator ; sumTextInstanceID)]

Use this function to start the sumText calculation.

Parameters

separator	the text to be inserted between each sumText result
sumTextInstanceID	please leave empty.

Returned result

the result is not relevant, as this function works together with the 2 other functions.

Special considerations

When no parameter is specified it defaults to "|".
NEW: It is now also possible to have a SumText without separators. Specify "-NoSeparator" as the parameter in this function.

Example usage

See "Using sumText" for a detailed instruction on how to use sumText.

TrText_UniqueLines

Syntax result = TrText_UniqueLines(switches ; theText)

Returns all the unique lines from theText. You can use this function in calculation field definitions or in script calculations.

Parameters

switches	(optional) determine way the result is returned
theText	the text for which the unique lines have to be found

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

All unique lines from theText.

Special considerations

This function ignores the case of the lines while comparing. The line that is included has the same case as the first line that's compared.

Example usage

Set Field [result ; TrText_UniqueLines("-Unused" ; theText)]

Say text field "theText" consists of the following lines:

this is a line
this is another line
These are the same
THESE are the same

Then result will be:

this is a line
this is another line
These are the same

Example 2

Text field "Text1" consists of the following lines:

12345
34567
12345
12345
34567

The result of UniqueLines will be:

12345
34567

TrText_UniqueWords

Syntax result = TrText_UniqueWords(switches ; theText)

Returns all unique words from theText. They are listed in order of appearance, and separated by a return. You can use this function in calculation field definitions or in script calculations.

Parameters

switches	(optional) determine way the result is returned
theText	the text for which the unique words have to be found

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

All unique words from theText, listed in order of appearance and separated by a return.

Special considerations

This function ignores the case of the words while comparing. The words returned have the first letter in upper case, the rest lower case (just like Paste from index in FileMaker Pro does).

Example usage

Set Field [result ; TrText_UniqueWords("-unused" ; theText)]

Say text field "theText" consists of the following lines:

this is a line
this is another line

Then the result will be:

This
Is
A
Line
Another

Example 2

If you want a sorted list of all words in a field you can use this calculation:

sortedWords = TrText_SortLines(TrText_UniqueWords("-Unused" ; theText))

TrText_Version

Syntax result = TrText_Version(switches)

Use this function to see which version of the plug-in is loaded.
Note: This function is also used to register the plug-in.

Parameters

switches determine the behaviour of the function

switches can be one of this:

-GetVersionString	the version string is returned (default)
-GetVersionNumber	returns the version number of the plug-in
-ShowFlashDialog	shows the Flash Dialog of the plug-in (returns 0)

If you leave the parameter empty the version string is returned.

Returned result

The function returns "" if this plug-in is not loaded. If the plug-in is loaded the result depends on the input parameter. It is either a:

VersionString:
If you asked for the version string it will return for example "Troi Text Plug-in 2.7"

VersionNumber:
If you asked for the version number it returns the version number of the plug-in x1000. For example version 2.7.1 will return number 2710.

ShowFlashDialogResult:
This will show the flash dialog and then return the error code 0.

Special considerations

IMPORTANT Always use this function to determine if the plug-in is loaded. If the plug-in is not loaded use of external functions may result in data loss, as FileMaker will return an empty field to any external function that is not loaded.

Example usage

TrText_Version will return "Troi Text Plug-in 2.8.1".

Example 2

TrText_Version("-GetVersionNumber") will return 2700 for version 2.7.
TrText_Version("-GetVersionNumber") will return 2701 for version 2.7b1
TrText_Version("-GetVersionNumber") will return 2730 for version 2.7.3

To use a feature introduced with version 2.7 test if the result is bigger than 2700.

TrText_XML

Syntax Set Field [result ; TrText_XML(switches ; nodeSpec ; XMLData)]

Returns the requested parts from text formatted as XML.

Parameters

switches	determine which data is returned
nodeSpec	the node you want data from
XMLData	the text which contains XML formatted data

Switches can be one of this:

-GetNode	return the node data
-GetAttributes	return the node attributes

Specifying Nodes:
Child nodes are specified from the root of the node downwards, and are separated by a slash: '/'. For example:

abc/def/ghi

If the XML contains multiple occurrences of a node you can retrieve a specific node by adding a number like this:

abc/def[2]/ghi[3]

This will get the 3rd 'ghi' node from the 2nd 'def' node of abc.

You can add spaces to the node specification to improve readability.

Returned result

The XML data requested, either the node or the attributes of the node.

Special considerations

The XML parsing function of Troi Text Plug-in has some known limitations:

No XML validation
Parsing will be only correct with a well-formed XML document (or a part thereof). If this is not the case the plug-in will try to parse it anyway, but the result may be unpredictable. The plug-in does not validate the XML.

Limited attribute retrieval
The plug-in can only return all attributes of a node at once. From this you have to get the individual attributes out yourself. An easy function to get a specified attribute out is on our own wish list. For the moment you can use the text functions in FileMaker to extract these from the "Attribute data" field.

Processing instructions and comments are not returned
The plug-in can't return processing instructions like "<?xml version="1.0" ?>" and comments. We think there is little need for this functionality. You can use the text functions in FileMaker Pro to extract these from the "XML data" field.

64000 character limit
The total length of a parameter of a plug-in function can only be 64000 characters. This is a limit of FileMaker Pro. Note that you can use Troi File Plug-in to extract portions of an external XML file.

Example usage

TrText_XML("-GetNode" ; "transaction/product/name" ; XML data)

Example 2

Say the field XML data contains this text:

```
<?xml version="1.0"?>
<xsd>
   <vendor id="12" >Troi Automatisering</vendor>
   <product>Time machine 1.0</product>
   <product>Hit Maker</product>
</xsd>
```

Then:
TrText_XML("-GetNode" ; "xsd/vendor" ; XML data)
returns 'Troi Automatisering'.
TrText_XML("-GetAttributes" ; "xsd/vendor" ; XML data)
returns 'id="12" '.
TrText_XML("-GetNode" ; "xsd/vendor/product[2]" ; XML data)
returns 'Hit Maker'.

TrText_XORText

Syntax result = TrText_XORText(switches ; text1 ; text2)

Returns all lines that are NOT both in text1 and text2.

Parameters

switches	(optional) determine way the result is returned
text1	first text
text2	second text

Switches can be empty or:

-ReturnAtEnd	add a return character at the end
-Unused	use this to make clear switches are not used, you can also use ""

Returned result

All lines that are NOT both in text1 and text2.

Special considerations

This is one of the 3 set manipulation functions.

Example usage

Set Field [result ; TrText_XORText(Text1 ; Text2)]

If Text1 contains:
AA
BB
and Text2 contains:
BB
CC
then the result will be:
AA
CC

Example 2

Text field "Text1" consists of the following lines:
1
10
12
22

And field "Text2" consists of the these lines:
1
2

The result of XORText will be:
2
10
12
22