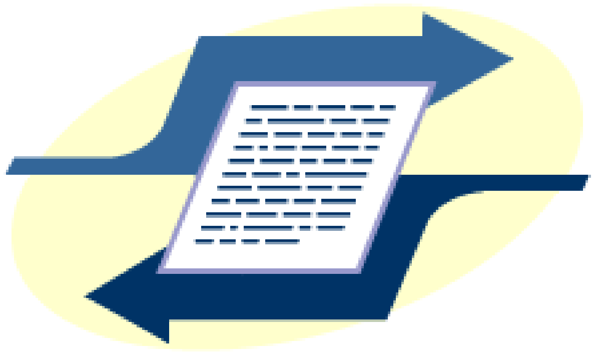


TROI CODING PLUG-IN™ 1.5 USER GUIDE

December 1999

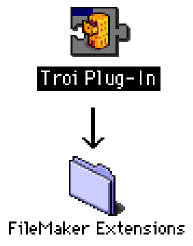
Troi Automatisering
Vuurlaan 18
2408 NB Alphen a/d Rijn
The Netherlands
Tel: +31 172-426606 Fax: +31-172-470539
You can also visit the Troi web site at: <http://www.troi.com/> for additional information.

Table of Contents

Installing plug-ins

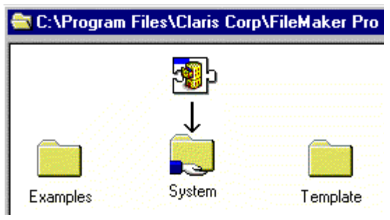
For Macintosh:

- Quit FileMaker Pro.
- Put the file "Troi Coding Plug-in" from the folder "MacOS Plug-in" into the "FileMaker 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 Coding Plug-In” 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 Coding Plug-in is used it will display a dialog box, indicating that it is loading and showing the registration status.



For Windows:

- Quit FileMaker Pro.
- Put the file "coding.fmx" from the directory "Windows" into the "SYSTEM" 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 'coding.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The Troi Coding Plug-in will display a dialog box, indicating that it is loading and showing the registration status.



TIP You can check which plug-ins you have loaded by going to the plug-in preferences: Choose **Preferences** from the **Edit** menu, and then choose **Plug-ins**.

You can now open the file "CodeExpl.fp3" to see how to use the plug-in's functions. There is also a Function overview available.

IMPORTANT There is a problem in FileMaker Pro 4.0v1. Please make sure that all plug-ins that are in the folder "FileMaker Extensions" are enabled in the preferences. (Under Edit/ Preferences/ Application/ Plug-ins). Make sure all plug-ins have a cross before their name. Remove plug-ins you don't use from the "FileMaker Extensions" folder. This bug is fixed in version 4.0v2, 4.1 and 5.0.

If You Have Problems

This user manual 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. 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 mistake 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 website: <html://www.troi.com/software/oserrrs.html>. This free FileMaker database lists all error codes for Windows and Mac OS!

Summary of functions

Plug-ins add new functions to the standard functions that are available in FileMaker Pro. You can see those extra functions for all plug-ins at the top right of the Specify Calculation Box:

IMPORTANT In the United States, commas act as list separators in functions. In other countries semi-colons might be used as list separators. The separator being used depends on the operating system your computer uses, as well as the one used when the file was created. All examples show the functions with comma's.

The Troi Coding Plug-in adds the following functions:

function name	short description
TrCo-Version	Check for correct version of the plug-in
Troi-Compress	Compresses text using a ZLIB algorithm.
Troi-Decompress	Decompresses text that was previously compressed.
Troi-Encrypt	Encrypts text using a DES algorithm and the current crypt key
Troi-Decrypt	Decrypts text using a DES algorithm
Troi-Set Crypt Key	Specify which key is used to encrypt and decrypt a text.
Troi-Code	Performs an encoding or decoding of a text field, depending on switches.
Troi-EncodeSafeAscii	Encodes a text to lower ASCII characters in the range 45...127.
Troi-EncodeShortSafeAscii	Encodes a text to Ascii characters in the range 45...127.
Troi-DecodeSafeAscii	Decodes a text in the SafeASCII format to the original text.
Troi-NumToBinary	Converts a number to its binary representation.
Troi-BinaryToNum	Converts a binary number to its decimal representation.
Troi-Checksum	Sum of the ASCII values of the characters modulo 1024.
Troi-TextSignature	Generates a signature of the characters that you can see.
Troi-Rotate13	Shifts the character values by 13

Using external functions

External functions for this plug-in can be used in a script step using a calculation. The functions can also be used in a define field calculation.

Function Reference

TrCo-Version

Syntax Set Field [result, External(TrCo-Version, "")]

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

no parameters, leave empty for future use.

Returned result

The name and version number of the loaded plug-in.
The function returns "" if this plug-in is not loaded.

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

External(TrCo-Version, "") will return as string like this: "Troi Coding Plug-in 1.5".

Troi-BinaryToNum

Syntax Set Field [result, External["Troi-BinaryToNum", "binaryNumber")]

Converts a binary number to its decimal representation.

Parameters

binaryNumber: the number that needs to be converted to its decimal representation.

Returned result

A decimal number

Example Usage

Set Field [result, External["Troi-BinaryToNum", "10")] will return as result 2

Set Field [result, External["Troi-NumToBinary", "10010")] will return as result 18

Troi-Checksum

Syntax Set Field [result, External["Troi-Checksum", "text")]

Sum of the ASCII values of the characters modulo 1024. ALL characters are counted, also non-printing characters like spaces and returns.

Parameters

text: *the text to calculate the checksum for.*

Returned result

a number

Special Considerations

A checksum might be the same for 2 different texts. The chance on this is normally quite low (one on 1024).

Example Usage

Set Field [result, External["Troi-Checksum", "Hello world.")] will give a result of 106.

You can use this function to see if the contents of a field has changed. You store the checksum and then later compare it to the current checksum.

Troi-Code

Syntax Set Field [result, External("Troi-Code", "switches | encryptionKey | text")]

Performs a encoding or decoding of the text field, depending on switches.

Parameters

switches	*specify what (de)coding action to perform*
encryptionKey	*the key to use*
text:	*the text to perform the action on*

switches can be:

-encryptDES	*encrypt using the DES algorithm (and the encryptionKey)*
-decryptDES	*decrypt using the DES algorithm (and the encryptionKey)*

Other switches are (not yet) possible.

Returned result

the coded text. This can be encrypted text or decrypted text.

Special Considerations

Use this function to encrypt and decrypt without a script.

LIMITATION

The total length of the parameters (= length of the switches + encryptionKey + text) should not be longer than 64000, as FileMaker will not call the "Troi-Code" function.) Practical this means that the text field should not be longer than about 63970 characters.

Example Usage

```
Set Field [ secretField,
        External("Troi-Code", " -encryptDES|" & gDecryptionKey & "|" & textField)]

Set Field [ result,
        External("Troi-Code", " -decryptDES|" & gDecryptionKey & "|" & secretField)]
```

gDecryptionKey = a global text field where the user can type in the key.

Troi-Compress

Syntax Set Field [result, External["Troi-Compress", "text")]

Compresses text using a ZLIB algorithm.

Parameters

text: *the text to compress*

Returned result

the compressed text string

Special Considerations

NOTE 1: short strings (less than 20 characters) of text might be longer after compression.
NOTE 2: the compression result can contain all ASCII codes (0-255). See "Troi-EncodeSafeAscii" for conversion to safe ASCII codes.

Example Usage

Set Field [result, External["Troi-Compress", "123456789 123456789 123456789")]
will result in the compressed string: "xú3426153∑∞T0ƒdWÿ"

EXAMPLE 2

In a document database you have defined a text field named "letterContents" which contains the main part of a letters.
Then you can define a calculation field:

letterCompressCalc calculation = External["Troi-Compress", letterContents)]

this field will contain the compressed version of the field.

Troi-DecodeSafeAscii

Syntax Set Field [result, External["Troi-DecodeSafeAscii", "text")]

Decodes a text in the SafeASCII format to the original text.

Parameters

text: the text to decode.

Returned result

the orginal text
OR
"$$-301 (Decoding Error)" , when the decoding failed.

Special Considerations

See also: Troi-EncodeSafeAscii and Troi-EncodeShortSafeAscii

Example Usage

```
Set Field [ result, External["Troi-DecodeSafeAscii",
  "this text is ignored!!!
    %Troi SafeAscii v1.0
     .V-PDon/Tt-Pforget-Pto-Phave-Pfun-\-PG.Pnther-Pand-PB/>rg-Q-P
   %End SafeAscii v1.0
   this text too...")]
```

gives this result: "• Don’t forget to have fun, Günther and Børg! "

EXAMPLE 2

In a database you have defined a text field named "receivedEmail" which contains the body of an email which contains a part that is encoded as ASCII Safe. Then you can define a calculation field:

```
    decodedCalc   calculation     = External[""Troi-DecodeSafeAscii",", receivedEmail)]
```

this field will contain the decoded text.

Troi-Decompress

Syntax Set Field [result, External["Troi-Decompress", "text")]

Decompresses text that was previously compressed.

Parameters

text: the text to decompress

Returned result

the decompressed text
OR
"$$ Decompression Error" + an error code, when the decompression failed.

Special Considerations

NOTE 1: this function can only decompress text that was previously compressed with "Troi-Compress" function (ZLIB algorithm). Currently no other algorithms like ZIP and Stuffit (.sit) are supported.

Example Usage

Set Field [result, External["Troi-Decompress", "xú3426153∑∞T0ƒdWÿ")]
will result in the decompressed string: "123456789 123456789 123456789"

EXAMPLE 2

In a document database you have defined a text field named "letterCompressed" which contains the main part of a letters, compressed. Then you can define a calculation field:

```
letterContentsCalc   calculation    Unstored, = External["Troi-Decompress", letterCompressed)]
```

this field will contain the uncompressed version of the contents.

Troi-Decrypt

Syntax Set Field [result, External["Troi-Decrypt", "text")]

Decrypts text using a DES algorithm and the current crypt key. Specify the correct key first with the function "Troi-Set Crypt Key".

Parameters

text: the text to decrypt

Returned result

the decrypted text

Special Considerations

If the current key does not match the key used to encrypt the text is not decrypted, and the input text is returned unchanged.

Example Usage

```
Set Field [ gErrorCode, External["Troi-Set Crypt Key", "mySecret" )]
If [gErrorCode = 0]
   Set Field [ result, External["Troi-Decrypt", "l’ùé—JtO<=! Û¡\}0Óÿ„]˘¿Cºdè≈˘")]
Endif
```

gives this result: "Hello World".

Troi-EncodeSafeAscii

Syntax Set Field [result, External["Troi-EncodeSafeAscii", "text")]

Encodes a text to lower ASCII characters in the range 45...127. The result can be sent safely over the Internet without any characters being changed. This function formats the output so that it is better readable for email.

Parameters

text to encode

Returned result

The result will be formatted like this:

```
%Troi SafeAscii v1.0
safe text line 1
safe text line 2
 …
%End SafeAscii v1.0
```

Special Considerations

You can decode it with: "Troi-DecodeSafeAscii".

NOTE: If you don't want formating use "Troi-EncodeShortSafeAscii" function.

Example Usage

Set Field [result, External["Troi-EncodeSafeAscii", "• Don’t forget to have fun, Günther and Børg! ")]

gives this result:

```
 %Troi SafeAscii v1.0
 .V-PDon/Tt-Pforget-Pto-Phave-Pfun-\-PG.Pnther-Pand-PB/>rg-Q-P
%End SafeAscii v1.0
```

Troi-EncodeShortSafeAscii

Syntax Set Field [result, External["Troi-EncodeShortSafeAscii", "text")]

Encodes a text to Ascii characters in the range 45...127. These character can be exported as tab separated text and also sent safely over the internet.

Parameters

text: text to encode

Returned result

The result will be formatted like this:

%Bencodedsafe text%E

This result of this function can be safely exported to for example TAB separated text.

Special Considerations

See also:
Troi-EncodeSafeAscii and Troi-DecodeSafeAscii

Example Usage

Set Field [result, External["Troi-EncodeShortSafeAscii", "• Don’t forget to have fun, Günther and Børg! ")]

gives this result: "%B.V-PDon/Tt-Pforget-Pto-Phave-Pfun-\-PG.Pnther-Pand-PB/>rg-Q-P%E"

EXAMPLE 2

In a database you have defined a text field named "patientName" and "patientData" which contains user data. Then you can define a calculation field:

```
safeNameCalc   calculation    = External[""Troi-EncodeShortSafeAscii",", patientName)]
safeDataCalc   calculation    = External[""Troi-EncodeShortSafeAscii",", patientData)]
```

these fields will contain the encoded text. If you export these safe fields you can get it like this:

```
Bj/>rn<TAB>Broken heel
```

TIP Use this function for sending encrypted data.

Troi-Encrypt

Syntax Set Field [result, External["Troi-Encrypt", "text")]

Encrypts text using a DES algorithm and the current crypt key. Specify a key first with the function "Troi-Set Crypt Key".

Parameters

text: the text to encrypt

Returned result

the encrypted text

Special Considerations

Be sure to remember the key (case sensitive!): without it you can not retrieve the original text.
Use "Troi-Code" to encrypt without the need for a script.

Example Usage

```
Set Field [ gErrorCode, External["Troi-Set Crypt Key", "mySecret" )]
If [gErrorCode = 0]
   Set Field [ result, External["Troi-Encrypt", "Hello World")]
   Set Field [ gErrorCode, External["Troi-Set Crypt Key", "different Key" )]
Endif
```

gives this result: "l’ùé—JtO<=! Û¡\}0Óÿ„]˘¿Cºdè≈ˇ"

EXAMPLE 2

In a database you have defined a text field named "patientData" which contains illness data. Then define a calculation field:

```
     encryptedDataCalc    calculation   Unstored  = External["Troi-Encrypt", patientData)]
```

this fields will contain the encrypted text.

TIP Use the Troi-EncodeSafeAscii function for sending data safely over the internet.

Troi-NumToBinary

Syntax Set Field [result, External["Troi-NumToBinary", "number")]

Converts a number to its binary representation.

Parameters

number = the number that needs to be converted to a binary.

Returned result

The number in binary notation.

Special Considerations

The maximum number to be converted = 4294967295

Example Usage

Set Field [result, External["Troi-NumToBinary", 2)] will return as result "10"

Set Field [result, External["Troi-NumToBinary", 18)] will return as result "10010"

NOTE
This sample uses a negative number:

Set Field [result, External["Troi-NumToBinary", -2)] will return as result "11111111111111111111111111111110"
which is the ones complement of 2.

Troi-Rotate13

Syntax Set Field [result, External["Troi-Rotate13", "text")]

Very simple coding of text. Shifts the character values by 13 to encrypt text stored in a FileMaker field. The field may be decrypted by using Rotate13 again.

Parameters

text: the text to Rotate

Returned result

the text that was Rotated

Special Considerations

Use this only as a simple way to make reading difficult.

Example Usage

Set Field [result, External["Troi-Rotate13", "Hello World")]

gives this result: "Uryyb Jbeyq"

EXAMPLE 2

Set Field [result, External["Troi-Rotate13", "Uryyb Jbeyq")]

gives this result: "Hello World"

Troi-Set Crypt Key

Syntax Set Field [result, External["Troi-Set Crypt Key", "the_key")]

Specify which key is used to encrypt and decrypt a text.

Parameters

the_key: *the key is used to encrypt and decrypt a text.*

Use this before you use the function "Troi-Encrypt" or "Troi-Decrypt". The key has to be at least 6 characters long and is case sensitive.

Returned result

0 if the key was set succesfully.

Example Usage

```
Set Field [ gErrorCode, External["Troi-Set Crypt Key", "mySecret" )]
If [gErrorCode = 0]
   Set Field [ result, External["Troi-Encrypt", "Hello World")]
   Set Field [ gErrorCode, External["Troi-Set Crypt Key", "different Key" )]
Endif
```

gives this result: "l’ùé—JtO<=! Û¡\}0Óÿ„]˘¿Cºdè≈ˇ"

Note that after the encryption the key is set to a different one, to prevent subsequent use of it.

Troi-TextSignature

Syntax Set Field [result, External["Troi-TextSignature", "text")]

Generates a signature of the characters that you can see. This means that only characters a-z, A-Z and 0-9 are used to generate the signature. So adding non-printing characters like spaces and returns doesn't change the signature.

Parameters

text: the text to calculate the signature for

Returned result

Signature: a string of 24 characters.

The characters are all lower ASCII and therefore safe to send across the internet.

Example Usage

```
Set Field [ result, External["Troi-TextSignature",
                    "Here is a sample text that you can see the signature of." )]
```

gives this result: "Cqd5yentvR5TN9bYSHG2MKdZ"

EXAMPLE 2

You can use this function to check if the (meaning) of a text was not changed, by adding the signature to the message.

```
Send Message[ message & "¶Signature=" & External["Troi-TextSignature", message )]
```

At the receiving end you need these fields:

```
signaturePos        = Position(messageReceived, "¶Signature=", 1,1)
messageClean        = Left(messageReceived,  signaturePos)
signatureReceived   = Middle(messageReceived,  signaturePos+ 12, Length(messageReceived))
messageOK           = External["Troi-TextSignature", messageClean ) = signatureReceived
```