The Expressions plugin
PRINTED MANUAL

Expressions plugin

Printed: 3/6/2026

Publisher

AGG Software

Production

©2006-2026 AGG Software
https://www.aggsoft.com

Table of Contents

1 Introduction

The filter module "Expressions" for our data loggers (for example, Advanced Serial Data Logger or Advanced TCP/IP Data Logger) is an interpreter for quick and easy evaluation of expressions. It is a smart tool easy to use. The plugin supports different data types (string, date, time, decimal, float and boolean), arithmetic and boolean operators, multiple levels of brackets, built-in math, string, boolean functions, and user-defined variables.

It supports the following operators and functions:

Mathematics operations

- + : Addition
- - : Subtraction
- * : Multiplication
- / : Division
- ^ : Exponential (only positive numbers for the base)

Mathematics functions: ABS, ATAN, COS, EXP, LN, ROUND, SIN, SQRT, SQR, TRUNC, and many others;
String functions: COPY, REPLACE, POS, and many others;
Logical functions: AND, OR, XOR, etc.

2 System requirements

The following requirements must be met for "Expressions" to be installed:

Operating system: Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. The latest service pack for the corresponding OS is required.

Free disk space: Not less than 5 MB of free disk space is recommended.

Special access requirements: You should log on as a user with Administrator rights in order to install this module.

The main application (core) must be installed, for example, Advanced Serial Data Logger.

3 Installing Expressions

1. Close the main application (for example, Advanced Serial Data Logger) if it is running;
2. Copy the program to your hard drive;
3. Run the module installation file with a double click on the file name in Windows Explorer;
4. Follow the instructions of the installation software. Usually, it is enough just to click the "Next" button several times;
5. Start the main application. The name of the module will appear on the "Modules" tab of the "Settings" window if it is successfully installed.

If the module is compatible with the program, its name and version will be displayed in the module list. You can see examples of installed modules on fig.1-2. Some types of modules require additional configuration. To do it, just select a module from the list and click the "Setup" button next to the list. The configuration of the module is described below.

You can see some types of modules on the "Log file" tab. To configure such a module, you should select it from the "File type" list and click the "Advanced" button.

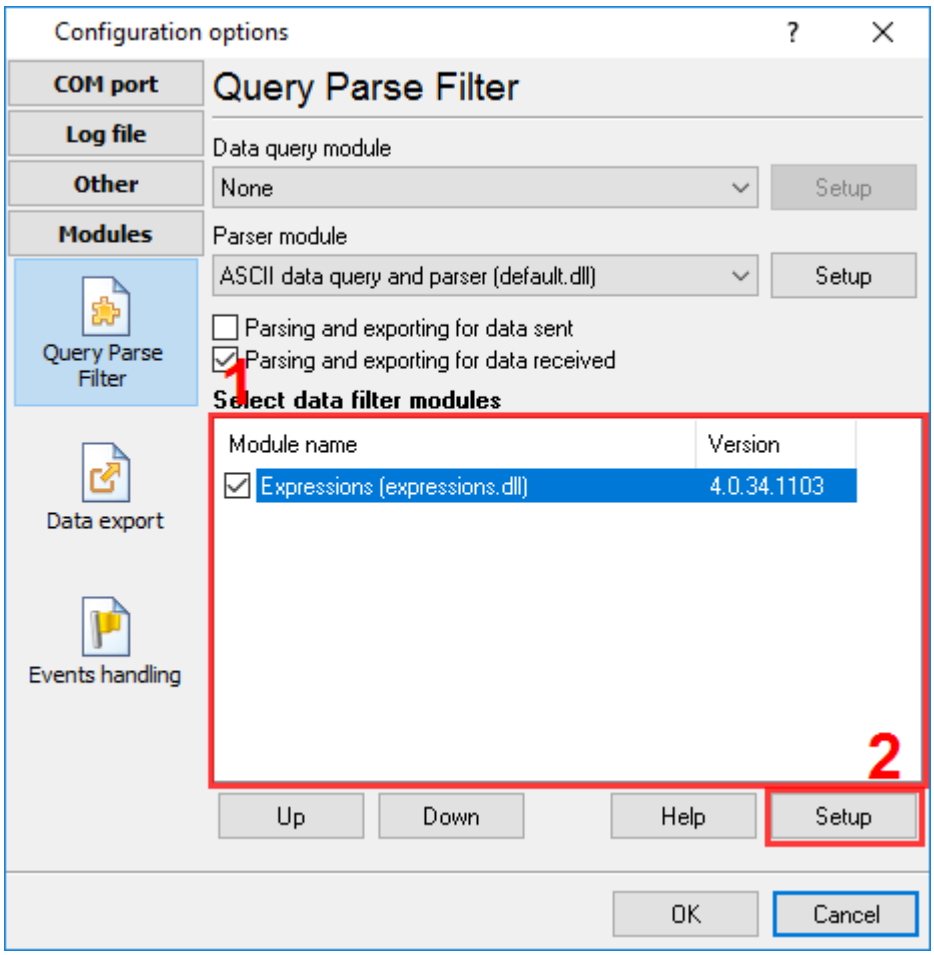


Fig. 1. Example of installed module

4 Glossary

Main program - it is the main executable of the application, for example, Advanced Serial Data Logger and asdlog.exe. It allows you to create several configurations with different settings and use different plugins.

Plugin - it is the additional plugin module for the main program. The plugin module extends the functionality of the main program.

Parser - it is the plugin module that processes the data flow, singling out data packets from it, and then variables from data packets. These variables are used in data export modules after that.

Core - see "Main program."

5 Configuration

The module configuration is very simple (pic.1). You may add one or more expressions to the edit window. One row contains one expression. Each expressions should have the following format:

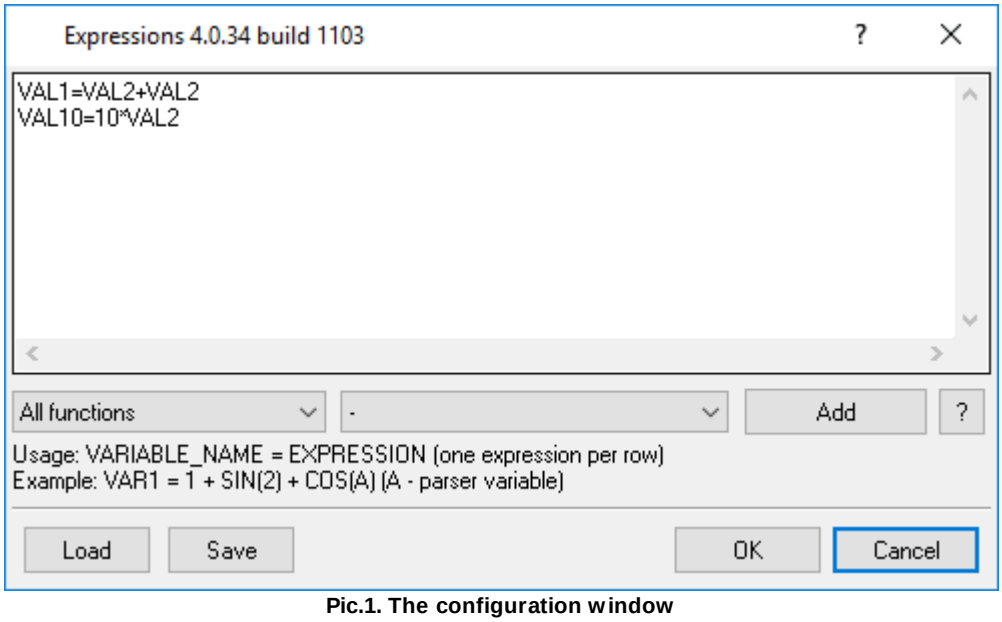
VARIABLE_NAME=EXPRESSION
Where VARIABLE_NAME is a name of the new or existing parser variable and EXPRESSION is any valid set of functions and/or variables.
This module may use all existing parser variables (you may select these variables in the "Variables" group). Your may read about parser variables in a parser's help file. Additionally you may define new variables here and assign a result of an expressions to these variables. Therefore user-defined variables may be used in data export modules as a regular parser variable.

All functions, operators and variables are exist in a drop-down list. If you want to add any function then you should:

1. Put the cursor to a position in the edit window;
2. Select the corresponding category (Date and Time for example);
3. Select a function that you need to insert to the edit window (DATE for example);
4. Click the "Add" button.

Later you may type your expressions directly, without using of drop-down lists.

All supported functions are described in following sections.



Pic.1. The configuration window

6 Supported functions

6.1 Constants

TRUE - boolean true.
FALSE - boolean false.

6.2 Math functions

ABS(X) - Abs returns the absolute value of the argument, X. X is an integer-type or real-type expression.

ARCCOS(X) - ArcCos returns the inverse cosine of X. X must be between -1 and 1. The return value is in the range [0..Pi], in radians.

ARCCOSH(X) - ArcCosh returns the inverse hyperbolic cosine of X. The value of X must be greater than or equal to 1.

ARCSIN(X) - ArcSin returns the inverse sine of X. X must be between -1 and 1. The return value will be in the range [-Pi/2..Pi/2], in radians.

ARCSINH(X) - ArcSinh returns the inverse hyperbolic sine of X.

ARCTAN2(X) - ArcTan2 calculates ArcTan(Y/X), and returns an angle in the correct quadrant. The values of X and Y must be between –2^64 and 2^64. In addition, the value of X can't be 0. The return value will fall in the range from -Pi to Pi radians.

ARCTANH(X) - ArcTanh returns the inverse hyperbolic tangent of X. The value of X must be between -1 and 1 (inclusive).

CEIL(X) - Call Ceil to obtain the lowest integer greater than or equal to X. The absolute value of X must be less than MaxInt.
For example:
Ceil(-2.8) = -2
Ceil(2.8) = 3
Ceil(-1.0) = -1

CLIP(X,Min,Max) - Returns Min if X<=Min, returns Max if X>=Max, otherwise returns X. For example:
CLIP(2, 3, 4) = 2
CLIP(3, 2, 4) = 3
CLIP(4, 2, 3) = 3

COS(X) - Use the COS to calculate the cosecant of X, where X is an angle in radians.

COSH(X) - Use the COSH to calculate the hyperbolic cosecant of X, where X is an angle in radians.

COTAN(X),

COTG(X) - Call Cotan or Cotg to obtain the cotangent of X. The cotangent is calculated using the formula 1 / Tan(X).

DEG(X) - Use Deg to convert angles measured in radians to degrees, where degrees = radians(180/pi).

EXP(X) - Exp returns the value of e raised to the power of X, where e is the base of the natural logarithms.

FLOOR(X) - Call Floor to obtain the highest integer less than or equal to X. For example:
Floor(-2.8) = -3
Floor(2.8) = 2
Floor(-1.0) = -1
Note: The absolute value of X must be less than MaxInt.

FRAC(X) - Frac function returns the fractional part of the argument X. X is a real-type expression. The result is the fractional part of X; that is, Frac(X) = X - Int(X).

HEX(X) - Converts a X value to a hexadecimal representation.

LN(X) - Ln returns the natural logarithm (Ln(e) = 1) of the real-type expression X.

LOG(Base, X) - Log returns the log base Base of X.

POW(Base, Exponent), POWER(Base, Exponent) - Power raises Base to any power. For fractional exponents or exponents greater than 65535, Base must be greater than 0.

POWLN2(X) - Power raises Ln2 base to any power. For fractional exponents or exponents greater than 65535, Base must be greater than 0.

RAD(X) - Use DegToRad to convert angles expressed in degrees to the corresponding value in radians, where radians = degrees(pi/180).

RANDOM(X) - Random returns a random number within the range 0 <= X < Range. If Range is not specified, the result is a real-type random number within the range 0 <= X < 1.

Note: Because the implementation of the Random function may change between compiler versions, we do not recommend using Random for encryption or other purposes that require reproducible sequences of pseudo-random numbers.

ROUND(X) - the Round function rounds a real-type value to an integer-type value. X is a real-type expression. Round returns an Int64 value that is the value of X rounded to the nearest whole number. If X is exactly halfway between two whole numbers, the result is always the even number. This method of rounding is often called "Banker's Rounding".

SIGN(X) - Use Sign to test the sign of a numeric value. Sign returns

0 if AValue is zero.
1 if AValue is greater than zero.
-1 if AValue is less than zero.

SIN(X) - Use the SIN to calculate the sine of X, where X is an angle in radians.

SINH(X) - Sinh calculates the hyperbolic sine of X.

SQR(X) - the Sqr function returns the square of the argument. X is a floating-point expression. The result, of the same type as X, is the square of X, or X*X.

SQRT(X) - X is a floating-point expression. The result is the square root of X.

TAN(X), TG(X) - Tan or Tg returns the tangent of X. Tan(X) = Sin(X) / Cos(X).

TRUNC(X) - the Trunc function truncates a real-type value to an integer-type value. X is a real-type expression. Trunc returns an Int64 value that is the value of X rounded toward zero.

6.3 Operators

- - subtraction

***** - multiplication

/ - real division

^ or ****** - Power raises base to any power. For fractional exponents or exponents greater than 65535, Base must be greater than 0. Example: x**y

+ - addition

< - less-than

<= - less-than-or-equal-to

<> - inequality

= - equality

> - greater-than

>= - greater-than-or-equal-to

AND - bitwise AND, Operand types: integer, Result type: integer, Example: X and Y

DIV - integer division. The value of x div y is the value of x/y rounded in the direction of zero to the nearest integer.

MOD - remainder. The mod operator returns the remainder obtained by dividing its operands. In other words, x mod y = x - (x div y) * y.

OR - bitwise OR, Operand types: integer, Result type: integer, Example: X or Y

SHL - bitwise shift left, Operand types: integer, Result type: integer, Example: X shl 2

SHR - bitwise shift right, Operand types: integer, Result type: integer, Example: X shr 2

XOR - bitwise XOR, Operand types: integer, Result type: integer, Example: X xor Y

6.4 String functions

SUBSTR(S, Index, Count), STRCOPY(S, Index, Count), COPY(S, Index, Count) - Returns a substring of a string. Copy returns a substring or subarray containing Count characters or elements starting at S[Index]. The substring is a unique copy (that is, it does not share memory with S, although if the elements of the array are pointers or objects, these are not copied as well.). If Index is larger than the length of S, Copy returns an empty string or array. If Count specifies more characters or array elements than are available, only the characters or elements from S[Index] to the end of S are returned.

FIRSTLINE(S) - Copies all characters until first CR or LF character.

TRIMLEFT(S), **LTRIM(S)** - Trims leading spaces and control characters from a string.

STRPOS(Substr, S), POS(Substr, S) - Pos searches for a substring, Substr, in a string, S. Substr and S are string-type expressions. Pos searches for Substr within S and returns an integer value that is the index of the first character of Substr within S. Pos is case-sensitive. If Substr is not found, Pos returns zero.

REMOVECHAR(S, Char) - Returns a string with occurrences of one character deleted. **REMOVECHAR** deletes occurrences of the character specified by **Char**. S is the source string, whose characters are deleted. The comparison operation is case sensitive.

REMOVENONPRINT(S) - Returns a string with occurrences of all non-printable characters (with ASCII code < 32) deleted.

REPLACE(S, OldPattern, NewPattern) - Returns a string with occurrences of one substring replaced by another substring. **REPLACE** replaces occurrences of the substring specified by OldPattern with the substring specified by NewPattern. S is the source string, whose substrings are changed. OldPattern is the substring to locate and replace with NewPattern. NewPattern is the substring to substitute for occurrences of OldPattern.The comparison operation is case insensitive.

REPLACECHAR(S, OldChar, NewChar) - Returns a string with occurrences of one substring replaced by another substring. **REPLACECHAR** replaces occurrences of the substring specified by OldChar with the substring specified by NewChar. S is the source string, whose substrings are changed. OldChar is the substring to locate and replace with NewChar. NewChar is the substring to substitute for occurrences of OldChar.The comparison operation is case sensitive.

TRIMRIGHT(S), RTRIM(S) - Trims trailing spaces and control characters from a string.

TRIM(S) - Trims leading and trailing spaces and control characters from a string.

6.5 Date and Time

DATE() - Returns current date (value of DateTime type).

DATE(S) - Returns a date value of DateTime type converted from the string S. The format of the string S - 'DD.MM.YYYY'. Example: DATE('15.01.2007')

DATE(Y,M,D) - Returns a date value of DateTime type converted from Y (year), M (month) and D (day) integer values. Example: DATE(2007, 1, 15)

DAY(X) - Returns a day part from the X value. The X value should be of DateTime type.

GOMONTH(X,Y) - adds Y months to X date value. Y may be negative or positive integer value. X should be of DateTime type.

MONTH(X) - Returns a month part from the X value. The X value should be of DateTime type.

NOW - Returns current date and time. The return value is with DateTime type.

TIME() - Returns current time (value of DateTime type).

TIME(S) - Returns a time value of DateTime type converted from the string S. The format of the string S - 'HH:NN'. Example: TIME('15:21')

TIME(H,M,S,MS) - Returns a date value of DateTime type converted from H (hour), M (minutes), S (seconds) and MS (milliseconds) integer values. Example: TIME(15, 21, 0, 0)

YEAR(X) - Returns a year part from the X value. The X value should be of DateTime type.

6.6 Miscellaneous functions

Conditional group

IIF(X,Y,Z) - if X boolean condition is True, then evaluates and returns Y, otherwise evaluates and returns Z.

NVL(X,Y) - if a variable with the X name does not exist, not defined or contains the NULL value, then the function will return Y.

DISCARD_DATA_PACKET_IF(X,Y) - if X boolean condition is True, then the program doesn't pass data packet to data filter plugins below the "Expressions" plugin and all data export plugins. Y is an optional text message to the program messages log.

> **Example**: Ignore all data packets if the parser variable with the "TEMPERATURE" is over the range: -20..20.
>
> DISCARD_DATA_PACKET_IF((TEMPERATURE>20)OR(TEMPERATURE<-20))
> DISCARD_DATA_PACKET_IF((TEMPERATURE>20)OR(TEMPERATURE<-20), "Value out of range")

GENERATE_EVENT_IF(X, Y, N1, V1, N2, V2) - if X boolean condition is True, then the program generates the Y event. The Y should be a string value. All data packet variables will be used as events parameters. Optional N1, V1 .. Nn, Vn value define additional event parameters.

If additional parameters contain EVENT-TO-CFG the program sends the event to the specified configuration. You may find the configuration name in the drop down box in the main window.

If additional parameters contain EVENT-GLOBAL the program sends the event to all configurations.

Example: Notify an user if the parser variable with the "TEMPERATURE" is over the range: -20..20.

```
GENERATE_EVENT_IF((TEMPERATURE>20)OR(TEMPERATURE<-20), 'TEMP_ALARM')
GENERATE_EVENT_IF((TEMPERATURE>20)OR(TEMPERATURE<-20), 'TEMP_ALARM',
'VAR_NAME', 'TEMPERATURE', 'VAR_VALUE', TEMPERATURE)
GENERATE_EVENT_IF((TEMPERATURE>20), 'TEMP_ALARM1', 'EVENT-TO-CFG', 'COM1')
GENERATE_EVENT_IF((TEMPERATURE>20), 'TEMP_ALARM2', 'EVENT-GLOBAL', TRUE)
```

Now, you should activate the "Events handling" plugin and add an event handler for the "TEMP_ALARM" event with necessary options.

SEND_EVENT_IF - this is another name of GENERATE_EVENT_IF.

REDIRECT_DATA_IF(X, Y) - if X boolean condition is True, then the program pass data packet to another configuration with Y, where this data packet will be process by all data filter and data export plugins. The data packet will be processed in the current configuration too. If you want to stop data processing in this configuration call DISCARD_DATA_PACKET_IF immediately after REDIRECT_DATA_IF.

Example:

```
REDIRECT_DATA_IF(VAR > 10, "COM2")
DISCARD_DATA_PACKET_IF(1=1)
```

SEND_BYTE_IF(X, Y) - if X boolean condition is True, then the program sends Y byte to the data source (COM port or TCP port). Note, some our loggers cannot send data.

SEND_DATA_IF(X, Y) - if X boolean condition is True, then the program sends Y **string**.

SEND_DATA_TO_DATA_SOURCE_IF(X, Z, Y) - if X boolean condition is True, then the program sends Y **string** to Z data source.

Example:

```
SEND_DATA_TO_DATA_SOURCE_IF(VAR > 10, "COM2", "Data string" + CHR(13) + CHR(10))
```

Please note, the configuration with the Z name must be added in the logger. This name must appear in the drop down list in the main window.

NEW_LOG_FILE_IF(X, Y) - if X boolean condition is True, then the program starts a new log file in the configuration Y. The program adds a sequential number to the current log file name.

Example:

NEW_LOG_FILE_IF(VAR > 10, "192.168.1.1:2000")

General group

MAX(A,B) - Call Max to compare two numeric values. Max returns the greater value of the two.

MIN(A,B) - Call Min to compare two numeric values in Delphi. Min returns the smaller value of the two.

SUM(A,B) - Equal for A+B, where A and B can be string or number.

Miscellaneous group

BYTETOSTR(X) - converts 1 byte in the array X to a unsigned byte value.

DOUBLETOSTR(X) - converts 8 bytes in the array X to a double value.

DOUBLETOSTRBE(X) - converts 8 bytes in "Big-endian" order in the array X to a double value.

INT64TOSTR(X) - converts 8 bytes in the array X to a 64 bit integer value.

INT64TOSTRBE(X) - converts 8 bytes in "Big-endian" order in the array X to a 64 bit integer value.

LONGINTTOSTR(X) - converts 4 bytes in the array X to a 32 bit integer value.

LONGINTTOSTRBE(X) - converts 4 bytes in "Big-endian" order in the array X to a 32 bit integer value.

LONGWORDTOSTR(X) - converts 4 bytes in the array X to a 32 bit unsigned decimal value.

LONGWORDTOSTRBE(X) - converts 4 bytes in "Big-endian" order in the array X to a 32 bit unsigned decimal value.

SINGLETOSTR(X) - converts 4 bytes in the array X to a float value.

SINGLETOSTRBE(X) - converts 4 bytes in "Big-endian" order in the array X to a float value.

SMALLINTTOSTR(X) - converts 2 bytes in the array X to a 16 bit signed decimal value.

SMALLINTTOSTRBE(X) - converts 2 bytes in "Big-endian" order in the array X to a 16 bit signed decimal value.

WORDTOSTR(X) - converts 2 bytes in the array X to a 16 bit unsigned decimal value.

WORDTOSTRBE(X) - converts 2 bytes in "Big-endian" order in the array X to a 16 bit unsigned decimal value.

6.7 Undocumented functions

Some function are not documented in this manual. You may find a description of these function in Google using the following keywords:

pascal "function name"

or

delphi "function name"