sbParser-SDK 3.62 for Win32

Creation and use of an sbParser object with ActiveX

sbParserX Properties

Get and Set

BSTR	Function
BSTR	VarDesc1
double	VarValue1
BSTR	VarDesc2
double	VarValue2
BSTR	VarDesc3
double	VarValue3
BSTR	VarDesc4
double	VarValue4
BSTR	VarDesc5
double	VarValue5
BSTR	VarDesc6
double	VarValue6
short	Base
short	AngularUnit

Get

double	Result
boolean	IsError
short	GlobalError
BSTR	UnknownFunction
OLE_HANDLE	hParser
BSTR	VersionNumber
short	InitCount

functions

double	ConvertBaseTod(**BSTR** *strValue, **short** BaseIn, **short** *errPos);
BSTR	dtochar(**BSTR** *strbuffer, **double** value, **short** point);
double	chartod(**BSTR** *strValue, **short** *errPos);
boolean	dConvertToBase(**BSTR** *strbuffer, **double** value, **short** b_point, **short** BaseIn);

csbparse.DLL

BSTR	GetParserVersionString(**BSTR** *strbuffer);
BSTR	GetParserVersionNumber(**BSTR** *strbuffer);

SoftBase@T-Online.de
http://home.t-online.de/home/SoftBase/serv02_f.htm

```
BSTR            GetParserVersionString(BSTR *strbuffer);
BSTR            GetParserVersionNumber(BSTR *strbuffer);
```

NAME

GetVersionString / GetVersionNumber

DESCRIPTION

Returns the version string or number of the *csbparse.DLL*.

PARAMETERS

BSTR* strbuffer - Points to the buffer that will receive the string.

RETURN VALUE

BSTR - The return value is the version string or number.

class **sbParser;**

NAME

sbParser (Overview)

DESCRIPTION

class sbParser (the creation and use of the ActiveX control sbParserX is explained here)

The class sbParser is an object for the calculation of functions of any length with a maximum of 24 variables.
In the function large and small letters are distinguished. Using subfunctions, constant factors and variables it must be considered, that e.g. sin(5) is recognized but SIN(5) is not. If only variable x is declared, X can not be interpreted.

SYNTAX

Possible subfunctions:

sin, **cos**, **tan**, **cot**,
asin, **acos**, **atan**, **acot**,
sinh, **cosh**, **tanh**, **coth**,
arsinh, **arcosh**, **artanh**, **arcoth**,
sqr (Square), **sqrt** (Square root),
exp, **lg**, **ln**,
sgn (Signum),
abs (absolute value),
round (Rounds E.g. round(5.8) = 6),
ceil (Rounds up E.g. ceil(5.8) = 6),
floor (Rounds down E.g. floor(5.8) = 5),
fac (Factorial E.g. fac(5) = 120),
rand (Random number in the field [0,c] E.g. rand(5) = 4),
GAMMA (The Gamma-function E.g. GAMMA(5.7) ~ 72.5276)
Annotation: The gamma function is implemented for values with one decimal place.

and
! (Factorial E.g. 5! = 120),
^ (Power E.g. 3^2 = 9),
Annotation: x^2 equals x^2 and x^3 equals x^3)
% (Percent E.g. 5+10% = 5.5 or 5*10% = 0.5)

Possible constant factors are:
pi, **e**.

Characters within **""** are ignored i.e. they are regarded as a comment.
E.g. in **"Costs 1996 are" 85.80"$"** only 85.80 is considered.

Possible brackets are:
(,) and **[,]** where no priority exists.

Abridging spelling can be used in case of variables which only consist of one character e.g. 3.175x, x(..) , 5.635(...).
The missing operator is always interpreted as *.

Additional subfunctions to built requests:

ispos	Value <= 0	Value > 0
isposq	Value < 0	Value >= 0
isneg	Value >= 0	Value < 0
isnegq	Value > 0	Value <= 0
isnull	Value != 0 (i.e. Value <> 0)	Value == 0
not (==isnull)	Value != 0	Value == 0

E.g.: ((Value1 < 3) AND (Value2 in [1;100])))
is equal to
(Value1-3 < 0) * (Value2-1 >= 0) * (Value2-100 <= 0),
because *****==AND, **+** == OR.
And this is finally equal to
isneg(Value1-3) * isposq(Value2-1) * isnegq(Value2-100)

1	**10**	1 (== true)
3	**1**	0 (== false)
-10	**-1**	0 (== false)

NAME

Function
VarDesc1, ..., VarDesc6
VarValue1, ..., VarValue6
Base
AngularUnit

DESCRIPTION

The sbParserX-properties to Get and Set.

PROPERTIES

BSTR Function - Points to a null-terminated string to be used as the **function** to work with.

BSTR VarDesc1, ..., VarDesc6 - Points to null-terminated strings specifying the names of the working variables.

double VarValue1, ..., VarValue6 - Specifies the values for the variables. E.g. VarValue1 is the value for variable VarDesc1, VarValue2 for VarDesc2, ... etc.

short Base - Specifies the number system the values given by the function string and the variables are based upon:

10 represents decimal,
2 represents binary,
16 represents hexadecimal,
8 represents octal.

short AngularUnit - Specifies the angular unit the trigonometric subfunctions are interpreted with.

1 sets angular unit to degree,
2 sets angular unit to radian,
3 sets angular unit to centesimal degree.

NAME

Result
IsError, GlobalError
UnknownFunction
hParser
VersionNumber
InitCount

DESCRIPTION

The sbParserX-properties to Get.

PROPERTIES

double Result - If the calculation is valid, this is the result of the calculation with the declared variables and given values.

boolean IsError- - Determines whether an error has occurred. If an error has occurred, IsError is true.
Use GlobalError to get extended error information.

short GlobalError- - Determines the sbParser object error code value. Possible errors are syntax [S]- and calculation errors [C].

[S]	-1	The function string is a value. <u>This is not an error!!!</u> (...but sometimes useful) E.g. 3.654 is a value, 3+6 is not.
[S][C]	0	No error has occurred.
[S]	1	The function string contains an unknown subfunction. (This subfunction can be determined by UnknownFunction.)
[S]	2	Incorrect brackets are set in the function string.
[S]	3	The function string contains an unknown constant factor or variable.
[C]	4	A multiplication causes an error.
[C]	5	A division causes an error.
[C]	6	The result of the calculation is not valid.
[C]	7	An addition causes an error.
[C]	8	A subtraction causes an error.
[S]	9	An invalid comment in the function string. See above **<u>sbParserX</u>** syntax for "".
[C]	10	The specified derivation is out of range.
[S]	11	Work was aborted by user.
[S]	12	The specified number of variables is not the number of the declared variables. If the number of variables exceeds the maximum this error value is set, too.
[S]	13	A +,-,* or / is set incorrectly in the function string.
[S]	14	The factorial function (!) is set incorrectly in the function string.
[S]	15	An unexpected end in the function string.
[S]	16	A subfunction is set incorrectly in the function string.

[S] 18 The function string contains an unknown character.

BSTR UnknownFunction - If an unknown subfunction error (see above GlobalError == 1) has occurred UnknownFunction determines the subfunction concerned.

OLE_HANDLE hParser - The return value is the HANDLE of the created sbParser object.

BSTR VersionNumber - See ***GetParserVersionNumber***.

short InitCount - Determines the number of the created sbParserX controls.

boolean **dConvertToBase** (**BSTR ***strbuffer, **double** value,
short b_point, **short** BaseIn);

NAME

dConvertToBase

DESCRIPTION

Converts a double value to a specified base and returns it as a string.

PARAMETERS

**BSTR ***strbuffer - Points to the buffer that will receive the string.

double value - Specifies the double value.

short b_point - Specifies the maximum number of digits for the result.

short BaseIn - Specifies the number system:

10	represents decimal,
2	represents binary,
16	represents hexadecimal,
8	represents octal.

RETURN VALUE

boolean - If the function succeeds, the return value is true. Otherwise it is false.

double **ConvertBaseTol**(**BSTR** *strValue, **short** BaseIn,
short *errPos);

NAME

ConvertBaseTod

DESCRIPTION

Converts a string representing a value of a specified base to a double.

PARAMETERS

BSTR * strValue - Points to a null-terminated string which identifies the value to be converted.

short BaseIn - Specifies the number system:
10 represents decimal,
2 represents binary,
16 represents hexadecimal,
8 represents octal.

short *errPos - If the scan of the string is successful errPos == -1. Otherwise errPos is the position of the character that stops the scan. (In C-Syntax: first character is 0, second is 1, ..., etc.)

RETURN VALUE

double - If the function succeeds, the return value is the result of the conversion.

BSTR **dtochar**(**BSTR** ***strbuffer**, **double** value,
short point);

NAME

dtochar

DESCRIPTION

Converts a double value into a string.

PARAMETERS

BSTR *strbuffer - Points to the buffer that will receive the string.

double value - Specifies the double value to be converted.

short point - Specifies the maximum number of digits for the result. If point is 0 the best number of digits will be used.

RETURN VALUE

BSTR - If the function succeeds, the return value is the result of the conversion.

double **chartod**(**BSTR** ***strValue, **short** ***errPos);)

NAME

chartod

DESCRIPTION

Converts a string representing a value to a double.

PARAMETERS

BSTR ***strValue - Points to a null-terminated string which identifies the value to be converted.

short ***errPos - If the scan of the string is successful errPos == -1. Otherwise errPos is the position of the character that stops the scan. (In C-Syntax: first character is 0, second is 1, ..., etc.)

RETURN VALUE

double - If the function succeeds, the return value is the result of the conversion.