XTRA for work with Microsoft Excel files

List of functions

new	XTRA initialisation.
registration	Registration of dmmXLS.x32.
cellProtection	Function of cell locking.
colRow2Ref	Transformation of cell in Col and Row format to Ref format.
deleteCol	The function deletes columns in the actual sheet in the xls document.
deleteRow	The function deletes rows in the actual sheet in the xls document.
errorDialog	A preselection giving choice to show the dialogue window with error statement.
errorLog	A preselection giving choice to save errors to log file.
errorMsg	The function gives information about an error occured while working with the XTRA.
firstCol	The function gives number of the first column containing data used in a sheet.
firstRow	The function gives number of the first row containing data used in a sheet.
getCellAnsiString	The function loads the contents of the cell in the xls file, whose format is UNICODE. It is returned as a value of the ANSI string type.
getCellBoolean	The function gives contents of a cell in boolean format.
getCellBooleanFormul	The function calculates the formula saved in the cell and it returns the result in the boolean format.
getCellDateTime	The function returns contents of the cell in the format DateTime as a propertyList.
getCellDouble	The function gives contents of a cell in double format.
getCellDoubleFormula	The function calculates the formula saved in the cell and it returns the result in the double format.
getCellHtml	The function returns the cell contents in Html format.
getCellHtmlRef	The function returns the cell contents in Html format.
getCellInteger	The function gives contents of a cell in integer format.
getCellIntegerFormula	The function calculates the formula saved in the cell and it returns the result in the integer format.
getCellRtf	The function returns the cell contents in Rtf format.
getCellRtfRef	The function returns the cell contents in Rtf format.
getCellString	The function gives contents of a cell in string format.
getCellStringFormula	The function calculates the formula saved in the cell and it returns the result in the string format.
getCellUTF8String	The function gives contents of a cell in UTF-8 string format.
getCellVal	The function returns contents of the cell according to the type of variable.
getCellValRef	The function returns contents of the cell according to the type of variable.
getSheetName	The function returns name of the actual sheet, set by the function setActiveSheet.
insertCol	The function adds columns to the actual sheet in the xls document.
insertRow	The function adds rows to the actual sheet in the xls document.
isNoError	Test made to show whether an error occured during work with the XTRA.
lastCol	The function gives number of the last column containing data used in a sheet.
lastRow	The function gives number of the last row containing data used in a sheet.
loadFromFile	Reading of an xls file.
loadFromHttp	Loading of the xls file from the http.
loadFromZip	Reading of xls documents saved in a zip archive.
ref2ColRow	Transformation of cell in Ref format to Col and Row format.
saveToFile	Xls file saving.
setActiveSheet	Setting of the actual sheet in xls,which we want to work with.
setCellBoolean	The function saves the contents in the boolean format to a cell in xls document.
setCellDateTime	The function saves contents of the propertyList to a cell in the xls document in the format DateTime.
setCellDouble	The function saves the contents in the double format to a cell in xls document.
setCellFontColor	The function sets colour of the text for the chosen cell.
setCellFontName	The function sets font for the chosen cell.
setCellFontSize	The function sets size of font for the chosen cell.
setCellFontStyle	The function sets a text style for a cell.
setCellFormula	The function saves the calculation formula in xls document.
setCellHAlign	The function sets text alignment in cell in the horizontal shape.

XTRA for work with Microsoft Excel files

List of functions

setCellInteger	The function saves the contents in the integer format to a cell in xls document.
setCellRotation	The function rotates a text in cell by a certain angle.
setCellString	The function saves the contents in the string format to a cell in xls document.
setCellUTF8String	The function saves the contents in the UTF-8 string format to a cell in xls document.
setCellVAlign	The function sets text alignment in cell in the vertical shape.
sheetProtection	Function of sheet locking.
xtraVersion	Finding out of version of XTRA dmmXLS.x32.
dataHttp	The event monitores the downloading of the xls file from the http.
xtraError	This event is invoked in case an error occurred during work with the XTRA.

XTRA for work with Microsoft Excel files

new
It is necessary to initiate the dmmXLS XTRA before the first use.

Example - Director
```
global xls
openXlib the pathName&"dmmXLS.x32"
xls=new(xtra "dmmXLS")

If the library dmmXLS.x32 is located in the XTRAS folder it is enough to insert
global xls
xls=new(xtra "dmmXLS")
```

Example - Authorware
```
xls:=NewObject("dmmXLS")
```

void=registration(name: string, code:string)
This function has to be called before the first use of the dmmXLS immediately after its initialisation. Unless the right registration name and number are inserted, an announcement "this is a demo version" will appear.

Parameters
Type of name is string, for the demo version name = "dmm".
Type of code is string, for the demo version code= "demo". The chain for commercial version is unique.
To register the user will receive parameters code and name.

Example - Director
```
global xls
xls.registration("dmm", "demo")
```

Example - Authorware
```
CallObject(xls; "registration"; "dmm"; "demo")
```

void=cellProtection(col: integer, row: integer, protect: boolean)
The function locks the cell.

Parameters
Col and row are coordinates of the cell we want to lock. Protect is a logical variable defining whether the cell is locked or open. For protect=true the cell is locked, for protect=false it is open.

Example - Director
```
xls.setActiveSheet(1)
xls.sheetProtection(false)
xls.setCellInteger(1, 5, 27)
xls.cellProtection(1, 5, false)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "sheetProtection"; false)
CallObject(xls; "setCellInteger"; 1; 5; 27)
CallObject(xls; "cellProtection("; 1; 5;  false)
```

string=colRow2Ref(col:integer, row:integer)
The function transforms Col and Row format defined cell to Ref format. Returned value is in string format. For example A1, C8, D22 etc.

Parameters
Col and row are coordinates of the cell we want to transform to Ref format. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
put xls.colRow2Ref(2,7)
```

Example - Authorware

```
CallObject(xls; "colRow2Ref" ; 2; 7)
```

void=deleteCol(col1:integer, col2:integer)

The function deletes columns in the actual sheet in the xls document.

Parameters

Col1 is a number of the column that we want to start deleting with. Col2 is a number of the column that we want to finish deleting with. Col1<=col2 is valid. If we delete the columns all the columns on the right will be shifted left.

Example - Director

```
global xls
xls.deleteCol(7,9)
```

Example - Authorware

```
CallObject(xls; "deleteCol" ; 7; 9)
```

void=deleteRow(row1:integer, row2:integer)

The function deletes rows in the actual sheet in the xls document.

Parameters

Row1 is number of the row that we want to start deleting with. Row2 is number of the row that we want to finish deleting with. Row1<=row2 is valid. If we delete the rows all the rows below will be shifted up.

Example - Director

```
global xls
xls.deleteRow(4,7)
```

Example - Authorware

```
CallObject(xls; "deleteRow" ; 4; 7)
```

void=errorDialog(dialog: boolean)

Using this function we can set whether a dialogue window, showing error statement appears at the moment of an error or not. This setting only functions in Autor mode and it simplifies the application debugging. If you decide not to set the dialogue window you can check the errors using the function errorMsg().

Parameters

Dialog is the only parameter of the function, whose type is boolean. If the value is true, the dialogue window with error statement is opened with every error in the XTRA. For the parameter value false the window is not shown. Default setting is dialog=false.

Example - Director

```
global xls
xls.errorDialog(true)
```

Example - Authorware

```
CallObject(xls; "errorDialog"; true)
```

void=errorLog(logFile: string, logSave: boolean)

Using this function we can set whether an error is saved to text log file at the time of the error or not. This setting only functions in Autor mode and it simplifies the application debugging.

Parameters

The function has two parametres. The parameter logFile represents name of the log file that we want to save the errors to. Type of the logSave parametre is boolean. If its value is true, every error in the XTRA is saved to log file. If its value is false nothing will be saved. Default setting is logSave=false.

Example - Director
```
global xls
xls.errorLog(the pathName&"log.txt", true)
```

Example - Authorware
```
CallObject(xls; errorLog; FileLocation ^ "log.txt"; true)
```

string=errorMsg()
The function gives text of the error that occured while working with dmmXLS. If everything went well, the function gives an empty chain

Parameters
There are no parametres in this function.

Example - Director
```
global xls
xls.loadFromFile(the pathName&"data.xls")
error=xls.ErrorMsg()
if error="" then val=xls.getCellString(2, 10)
```

Example - Authorware
```
error:=CallObject(xls; "errorMsg")
```

integer=firstCol()
The function gives number of the first column containing data used in a sheet.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
minCol=xls.firstCol()
```

Example - Authorware
```
minCol:=CallObject(xls; "firstCol")
```

integer=firstRow()
The function gives number of the first row containing data used in a sheet.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
minRow=xls.firstRow()
```

Example - Authorware
```
minRow:=CallObject(xls; "firstRow")
```

void=getCellAnsiString(col:integer, row:integer, codePage: integer)
The function loads the contents of the cell in the xls file, whose format is UNICODE. It is returned as a value of the ANSI string type.

Parameters
Col and row are coordinates of the cell, whose contents we want to read. The parameter codePage represents the code page to be used to convert the UNICODE from. In the UNICODE text, only signs included in the appropriate page code can be used. If any other signs are used, they will replaced by question marks.

CodePage - Language
874 - Thai
932 - Japan
936 - Chinese (PRC, Singapore)
949 - Korean
950 - Chinese (Taiwan, Hong Kong)
1200 - Unicode (BMP of ISO 10646)
1250 - Eastern European
1251 - Cyrillic
1252 - Latin 1 (US, Western Europe)
1253 - Greek
1254 - Turkish
1255 - Hebrew
1256 - Arabic
1257 - Baltic

Example - Director
```
global xls
xls.setActiveSheet(1)
val=xls.getCellAnsiString(2, 10, 1251)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellAnsiString"; 2; 10; 1251)
```

boolean=getCellBoolean(col:integer, row:integer)
The function reads contents of a cell in xls format and gives it back in boolean format.

Parameters
Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
xls.setActiveSheet(1)
val=xls.getCellBoolean(2, 10)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellBoolean"; 2; 10)
```

boolean=getCellBooleanFormula(col:integer, row:integer)
The function calculates the formula saved in the cell and it returns the result in the boolean format.

Parameters
Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
xls.setActiveSheet(1)
val=xls.getCellBooleanFormula(2, 10)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellBooleanFormula"; 2; 10)
```

propertyList=getCellDateTime(col:integer, row:integer)

The function loads the contents of the cell in xls document, whose format is DateTime, and it returns it in Abstract Data Type format, for example [#year:2002, #month:2, #day:2, #hour:8, #minute:27, #second:42].

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellDateTime(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellDateTime"; 2; 10)
```

double=getCellDouble(col:integer, row:integer)

The function reads contents of a cell in xls format and gives it back in string double.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellDouble(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellDouble"; 2; 10)
```

double=getCellDoubleFormula(col:integer, row:integer)

The function calculates the formula saved in the cell and it returns the result in the double format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellDoubleFormula(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellDoubleFormula"; 2; 10)
```

string=getCellHtml(col:integer, row:integer)

The function returns the cell contents in Html format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
put xls.getCellHtmll(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellHtml"; 2; 10)
```

string=getCellHtml(ref: string)

The function returns the cell contents in Html format.
This function is identical with the function getCellHtml, the only difference is that we enter one string parameter Ref instead of Col and Row parameters.

Parameters

Ref is a coordinate of the cell in format ColRow. Type of parametr is string, for example A1, C8, D22 etc.

Example - Director

```
global xls
xls.setActiveSheet(1)
put xls.getCellHtmlRef("B7")
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellHtmlRef"; "B7")
```

integer=getCellInteger(col:integer, row:integer)

The function reads contents of a cell in xls format and gives it back in integer format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellInteger(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellInteger"; 2; 10)
```

integer=getCellIntegerFormula(col:integer, row:integer)

The function calculates the formula saved in the cell and it returns the result in the integer format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellIntegerFormula(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellIntegerFormula"; 2; 10)
```

string=getCellRtf(col:integer, row:integer)

The function returns the cell contents in Rtf format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
put xls.getCellRtf(2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellRtf"; 2; 10)
```

string=getCellRtfRef(ref: string)

The function returns the cell contents in Rtf format.
This function is identical with the function getCellRtf, the only difference is that we enter one string parameter Ref instead of Col and Row parameters.

Parameters

Ref is a coordinate of the cell in format ColRow. Type of parametr is string, for example A1, C8, D22 etc.

Example - Director

```
global xls
xls.setActiveSheet(1)
put xls.getCellRtfRef("B7")
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellRtfRef"; "B7")
```

string=getCellString(col:integer, row:integer)

The function reads contents of a cell in xls format and gives it back in string format. If, for example, integer value is included in the cell, it is also converted into string format. If a formula is included in the cell, its value is given back.
In Director up to the version 10, this function returns ansi coded page. In Director 11 the function return text in UTF-8.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=xls.getCellString( 2, 10)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellString"; 2; 10)
```

string=getCellStringFormula(col:integer, row:integer)

The function calculates the formula saved in the cell and it returns the result in the string format.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
xls.setActiveSheet(1)
val=xls.getCellStringFormula(2, 10)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellStringFormula"; 2; 10)
```

string=getCellUTF8String(col:integer, row:integer)

The functions loads the contents of the cell in the xls file, whose format is UNICODE. It is returned as a value of the UTF-8 string type.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
xls.setActiveSheet(1)
val=xls.getCellUTF8String(2, 10)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellUTF8String"; 2; 10)
```

variant=getCellVal(col:integer, row:integer)

The function returns contents of the cell according to the type of variable. If integer value is saved in a cell, integer value will be returned too. If double value is saved in a cell, double value will returned etc. If a Director or Authorware incompatible value is saved in a cell, string value is returned.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow.

Example - Director
```
global xls
xls.setActiveSheet(1)
put xls.getCellVal(2, 10)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellVal"; 2; 10)
```

variant=getCellValRef(ref: string)

The function returns contents of the cell according to the type of variable. If integer value is saved in a cell, integer value will be returned too. If double value is saved in a cell, double value will returned etc. If a Director or Authorware incompatible value is saved in a cell, string value is returned.
This function is identical with the function getCellVal, the only difference is that we enter one string parameter Ref instead of Col and Row parameters.

Parameters

Ref is a coordinate of the cell in format ColRow. Type of parametr is string, for example A1, C8, D22 etc.

Example - Director
```
global xls
xls.setActiveSheet(1)
put xls.getCellValRef("B7")
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getCellValRef"; "B7")
```

string=getSheetName()
The function returns name of the actual sheet, set by the function setActiveSheet.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
xls.setActiveSheet(1)
put xls.getSheetName()
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
val:=CallObject(xls; "getSheetName")
```

void=insertCol(col:integer, colCount:integer)
The function adds empty columns to the actual sheet in the xls document.

Parameters
Col is a number of the column front which we want to add new columns. ColCount is a number of new columns that we want to add.

Example - Director
```
global xls
xls.insertCol(2,10)
```

Example - Authorware
```
CallObject(xls; "insertCol" ; 2; 10)
```

void=insertRow(row:integer, rowCount:integer)
The function adds rows to the actual sheet in the xls document.

Parameters
Row is a number of the row above which we want to add new rows. RowCount is a number of new rows that we want to add.

Example - Director
```
global xls
xls.insertRow(27,5)
```

Example - Authorware
```
CallObject(xls; "insertRow" ; 27; 5)
```

boolean=isNoError()
The function gives back true, if no error occured during an operation. If an error occurs the function gives back false.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
xls.loadFromFile(the pathName&"data.xls")
if xls.isNoError() then val=xls.getCellString(2, 10)
```

Example - Authorware
```
error:=CallObject(xls; "isNoError")
```

integer=lastCol()
The function gives number of the last column containing data used in a sheet.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
maxCol=xls.lastCol()
```

Example - Authorware
```
maxCol:=CallObject(xls; "lastCol")
```

integer=lastRow()
The function gives number of the last row containing data used in a sheet.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
maxRow=xls.lastRow()
```

Example - Authorware
```
maxRow:=CallObject(xls; "lastRow")
```

boolean=loadFromFile(file:string, password: string)
This function reads an xls file for next use. We cannot read the contents of cells, unless it si read using the function loadFromFile.
If xls or xlsx were read correctly the function returns true. If an error occurred, false is returned.

Parameters
Type of file is string and we insert name of an xls or xlsx file including the path. Type of password is string. We enter password in case we run an xls or xlsx file protected by a password.

Example - Director
```
global xls
if xls.loadFromFile(the pathName&"data.xls", "") =false then   ..... file is not protected by a password
   alert("error")
end if

if xls.loadFromFile(the pathName&"data2.xls", "abc") =false then
   alert("error")
end if
```

Example - Authorware
```
CallObject(xls; "loadFromFile"; FileLocation ^ "data.xls"; "")  ..... file is not protected by a password
CallObject(xls; "loadFromFile"; FileLocation ^ "data2.xls"; "abc")
```

boolean=loadFromHttp(url:string)
This function enables to download xls files from the internet using the method get and protocol http. In the case the file has been successfully downloaded the function gives true, otherwise in gives false. To have this function activated in the XTRA you need to have the additive library dmmHttp in the directory where dmmXLS is placed.

Parameters
The function has one parameter being url, whose type is string and it represents Url of the downloaded document.

Example - Director
```
global xls
xls.loadFromHttp("http://xtra.web-cd.net/data.xls")
```

Example - Authorware
```
CallObject(xls; "loadFromHttp"; "http://xtra.web-cd.net/data.xls")
```

boolean=loadFromZip(zipFile, zipPassword, xlsFile:string)

This function loads xls documents saved in a zip archive. The work procedure is easy. We prepare xls files and compress them to a zip. We can protect this zip with a password. The function loadFromZip enables to read separate files of the archive and to show them in Director or Authorware using XTRA dmmXLS. To have this function activated in the XTRA you need to have the additive library dmmZip in the directory where dmmXLS is placed. In the case the file has been successfully loaded the function gives true, otherwise it gives false.

Parameters
There are 3 parametres in this function. ZipFile is a zip archive, in which we keep the documents. ZipPassword is a password for the zip archive. If there is no password, we insert an empty chain. XlsFile is a name xls file in a zip document. We have to insert the name including the extension.

Example - Director
```
global xls
xls.loadFromZip(the pathName & "demo.zip", "abc", "data.xls")
```

Example - Authorware
```
CallObject(xls; "loadFromZip"; FileLocation ^ "demo.zip"; "abc"; "data.xls")
```

list=ref2ColRow(ref: string)

The function transforms Ref format defined cell to Col and Row format. Returned value is in Abstract Data Types format.
For example: [#Col:10,#Row:18]

Parameters
Ref is a coordinate of the cell in format ColRow. Type of parametr is string, for example A1, C8, D22 etc.

Example - Director
```
global xls
put xls.ref2ColRow("A22")
put xls.ref2ColRow("B22").col
put xls.ref2ColRow("B22").row
```

Example - Authorware
```
val:=CallObject(xls; "ref2ColRow" ; "A22")
```

void=saveToFile(file:string, password: string)

The function saves an xls file on the disc.

Parameters
Type of file is string and we insert name of the xls, including the path. Type of password is string. We enter password in case we save an xls or xlsx file protected by a password.

Example - Director
```
global xls
xls.saveToFile(the pathName&"data.xls", "")  ..... file is not protected by a password
xls.saveToFile(the pathName&"data.xls", "abc")
```

Example - Authorware
```
CallObject(xls; "saveToFile"; FileLocation ^ "data.xls"; "") ..... file is not protected by a password
CallObject(xls; "saveToFile"; FileLocation ^ "data.xls"; "abc")
```

void=setActiveSheet(sheet:integer)

The function sets the actual sheet in xls, which we want to work with.

Parameters

Sheet is number of the sheet in xls, which we want to work with. The parameter can reach the values 1 and higher, depending on how many sheets we define in xls.

Example - Director

```
global xls
xls.setActiveSheet(2)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 2)
```

void=setCellBoolean(col:integer, row:integer, val: boolean)

The function saves the content in the boolean format to a cell in xls document.

Parameters

Col and row are coordinates of the cell, that we want to save the string value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. Type of val is boolean and insert a value, that we save to a cell.

Example - Director

```
global xls
xls.setActiveSheet(1)
xls.setCellBoolean(2, 10, false)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellBoolean"; 2; 10; false)
```

void=setCellDateTime(col:integer, row:integer, val:propertyList)

The function saves DateTime value to a cell of xls document.The DateTime value is in Abstract Data Type format, for example [#year:2002, #month:2, #day:2, #hour:8, #minute:27, #second:42].

Parameters

Col and row are coordinates of the cell, that we want to save the DateTime value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow.
Type of val is propertyList and we insert it in the Abstract Data Type format.

Example - Director

```
global xls
xls.setActiveSheet(1)
val=[#year:2002, #month:2, #day:2, #hour:8, #minute:27, #second:42]
xls.setCellDateTime(2, 10, val)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellDateTime"; 2; 10; val)
```

void=setCellDouble(col:integer, row:integer, val: double)

The function saves the content in the double format to a cell in xls document.

Parameters

Col and row are coordinates of the cell, that we want to save the string value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. Type of val is double and insert a value, that we save to a cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellDouble(2, 10, 27.102)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellDouble"; 2; 10; 27.102)
```

void=setCellFontColor(col:integer, row:integer, fontColor: string)
The function sets colour of the text for the chosen cell.

Parameters
Col and row are coordinates of the cell, where we want set text colour. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. FontColor is defined in hexadecimal shape #RRGGBB. Example: #F0FF00.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellFontColor(2, 10, "#FF0000")
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellFontColor"; 2; 10; "#FF0000")
```

void=setCellFontName(col:integer, row:integer, fontName: string)
The function sets font for the chosen cell.

Parameters
Col and row are coordinates of the cell, where we want set font name. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. FontName is name of font we want to set for the chosen cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellFontName(2, 10, "Tahoma")
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellFontName"; 2; 10; "Tahoma")
```

void=setCellFontSize(col:integer, row:integer, fontSize: integer)
The function sets size of font for the chosen cell.

Parameters
Col and row are coordinates of the cell, where we want set font size. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. FontSize is size of font we want to set for the chosen cell. FontSize is defined in pixel.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellFontSize(2, 10, 17)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellFontSize"; 2; 10; 17)
```

void=setCellFontStyle(col:integer, row:integer, style: List)

The function sets a text style for a cell.

Parameters

Col and row are cell coordinates of the cell for which we set style. The type of style is Abstract Data Types format. In style we can insert the variables in the shape of the symbols that can have the following values #bold, #italic, #underline and #strikeout.
For example, setCellFontStyle(1, 1, [#bold, #italic]) sets bold and italic styles for a cell having coordinates[1,1], italic. setCellFontStyle(1, 1, [#bold]) only sets bold etc

Example - Director

```
global xls
xls.setActiveSheet(1)
xls.setCellVAlign(2, 10, setCellFontStyle, [#bold, #italic])
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellFontStyle"; 2; 10; [#bold; #italic])
```

void=setCellFormula(col:integer, row:integer, formuleStr: string)

The function saves the calculation formula in xls document.

Parameters

Col and row are coordinates of the cell, whose contents we want to read. Col is a column coordinate and row is a row coordinate. The values range from firstCol and lastCol, or firstRow and lastRow. FormuleStr is of the string type and we insert here our own calculation formula that we want to be saved in a cell of the xls document.

Example - Director

```
global xls
xls.setActiveSheet(1)
xls.setCellFormula(2, 10,, "SUM(C1:C4)")
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellFormula"; 2; 10; , "SUM(C1:C4)")
```

void=setCellHAlign(col:integer, row:integer, hAlign: symbol)

The function sets text alignment in cell in the horizontal shape.

Parameters

Col and row are coordinates of a cell for which we want to set horizontal alignment. Type of the parameter hAlign is symbol and it can reach the following values: #left, #center, and #right.

Example - Director

```
global xls
xls.setActiveSheet(1)
xls.setCellHAlign(2, 10, #center)
```

Example - Authorware

```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellHAlign"; 2; 10; #center)
```

void=setCellInteger(col:integer, row:integer, val: integer)

The function saves the content in the integer format to a cell in xls document.

Parameters
Col and row are coordinates of the cell, that we want to save the string value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. Type of val is integer and insert a value, that we save to a cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellInteger(2, 10, 27)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellInteger"; 2; 10; 27)
```

void=setCellRotation(col:integer, row:integer, angle: integer)
The function rotates a text in cell by a certain angle.

Parameters
Col and row are coordinates of a cell in which we want to rotate text. Angle is the angle, defined in degrees, by which we want to rotate the text in the cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellRotation(2, 10, 45)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellRotation"; 2; 10; 45)
```

void=setCellString(col:integer, row:integer, val: string)
The function saves the content in the string format to a cell in xls document.

Parameters
Col and row are coordinates of the cell, that we want to save the string value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. Type of val is string and insert a text chain, that we save to a cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellString(2, 10, "XTRA")
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellString"; 2; 10; "XTRA")
```

void=setCellUTF8String(col:integer, row:integer, val: string)
The function saves a value of the UTF-8 string. type to the the cell in the xls document. When saving in Excel, the values are automatically reconverted to the UNICODE format.

Parameters
Col and row are coordinates of the cell, that we want to save the string value to. Col is a coordinate of column and row is a coordinate of row. The values range from firstCol to lastCol, respectively from firstRow to lastRow. Type of val is unicode string and insert a text chain, that we save to a cell.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellUTF8String(2, 10, "XTRA")
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellUTF8String"; 2; 10; "XTRA")
```

void=setCellVAlign(col:integer, row:integer, vAlign: symbol)
The function sets text alignment in cell in the vertical shape.

Parameters
Col and row are coordinates of a cell for which we want to set vertical alignment. Type of the parameter vAlign is symbol and it can reach the following values: #top, #middle, and #bottom.

Example - Director
```
global xls
xls.setActiveSheet(1)
xls.setCellVAlign(2, 10, #middle)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "setCellVAlign"; 2; 10; #middle)
```

void=sheetProtection(protect: boolean)
The function locks the sheet.

Parameters
Protect is a logical variable defining whether the sheet is locked or open. For protect=true the sheet is locked, for protect=false it is open.

Example - Director
```
xls.setActiveSheet(1)
xls.sheetProtection(false)
```

Example - Authorware
```
CallObject(xls; "setActiveSheet" ; 1)
CallObject(xls; "sheetProtection"; false)
```

list=xtraVersion()
The function returns information about the XTRA dmmXLS. The function returns values in the format Abstract Data Types:
```
[#fileType: "Xtra (32)",
 #CompanyName: "Studio dmm",
 #FileDescription: "dmmXLS.x32",
 #FileVersion: "1.8.0.17",
 #InternalName: "dmmXLS.x32",
 #LegalCopyRight: "Â© Studio dmm 1992-2005",
 #LegalTradeMarks: "",
 #OriginalFileName: "dmmXLS.x32",
 #productName: "dmmXLS.x32",
 #productVersion: "1.8.0.0"]
```

The meaning of the items is clear and it is not necessary to describe it closer.

Parameters
There are no parametres in this function.

Example - Director
```
global xls
xv=xls.xtraVersion()
put "FileDescription:" && xv.FileDescription
put "FileVersion:" && xv.FileVersion
```

Example - Authorware

```
xv:=CallObject(xls; "xtraVersion")
Trace("FileDescription: " ^ xv[#FileDescription])
Trace("FileVersion: " ^ xv[#FileVersion])
```

dataHttp progress

This event (callback function) is invoked at the moment when the user starts to download the xls file from the http. It monitores the downloading of the xls file. The parameter progress represents how many percents have already been downloaded. The values ranges from 0 to 100.

Example - Director

```
on dataHttp progress
    put progress
end
```

xtraError err

This event is invoked in case an error occurred during work with the XTRA. The error is returned in the parameter err.

Example - Director

```
on xtraError err
    put err
end
```