dmmMDB - Xtra for Director and Authorware

List of functions

new	XTRA initialisation.
dbRegistration	Registration of XTRA dmmMDB.x32.
dbActiveSQL	Insertion and activation of an SQL command, whose type is SELECT.
dbAppend	Addition of a new record to the database.
dbBlobBMP2Member	Lecture of an image from a database and its saving to cast member.
dbBof	A test to check whether the actual record is the first one in the database.
dbClearBookmarks	All bookmarks deletion.
dbClose	Database file closure.
dbColumnCount	Number of fields in the database.
dbColumns	Returned values of database fields.
dbCompactAndRepair	This function corrects and comprimes the mdb file.
dbDelete	Cancellation of the actual record in the database.
dbDeleteBookmark	Bookmark deletion.
dbEdit	Change of item values of the database in the actual record.
dbEof	A test to check whether the actual record is the last one in the database.
dbErrorDialog	A preselection giving choice to show the dialogue window with error statement.
dbErrorLog	A preselection giving choice to save errors to log file.
dbErrorMsg	An error occured while working with mdb file.
dbExecSQL	Insertion and activation of an SQL command, whose type is INSERT, UPDATE or DELETE.
dbFilter	Setting of the filter for the database table.
dbFiltered	Switching of the filter for the database table.
dbFirst	A jump on the first record of the database.
dbGetADOVersion	Finding out of version of the database engine.
dbGetBookmarks	Information about what boomarks are setted.
dbGetFieldByName	A given value of the item defined by its name.
dbGetFieldByNumber	A given value of the item defined by its number in the database.
dbGetFields	Returned values of the actual record.
dbGetProducer	Returning of the formatted text using the pattern.
dbGetTableNames	Finding out of what tables and requests are included in the database mdb file.
dbGoToBookmark	A jump on an already create bookmark in the database.
dbIndexFieldNames	Setting of indexes for the table and sorting of the table in conformity with these indexes.
dbInsertBookmark	Placement of a bookmark on a certain record in the database.
dbIsFiltered	Testing of the situation of the filter.
dbIsNoError	Test made to show whether an error occured during work with the XTRA.
dbLast	A jump on the last record of the database.
dbLocate	Finding of the record fulfiling certain conditions in the database.
dbMember2BlobBMP	Saving of image cast member contents to database.
dbMoveBy	A jump n records ahead in the database.
dbNext	A jump on the next record of the database.
dbOpen	Database file opening.
dbOpenTable	Opening of the database table.
dbPrior	A jump on the previous record of the database.
dbRecordCount	Number of records of the database.
dbRecordNo	Record number of the actual record in the database.
dbSetProducer	Setting of the pattern for the database output.
dbXtraVersion	Finding out of version of XTRA dmmMDB.x32.

new

It is necessary to initiate the dmmMDB XTRA before the first use.
If the library dmmMDB.x32 is located in the XTRAS folder it is enough to insert
global mdb
mdb=new(xtra "dmmMDBJet")

Example - Director

```
global mdb
openXlib the pathName&"dmmMDB.x32"
mdb=new(xtra "dmmMDBJet")
```

Example - Authorware

```
mdb:=NewObject("dmmMDBJet")
```

void=dbRegistration(name: string, code:string)

This function has to be called before the first use of the dmmMDB 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 mdb
mdb.dbRegistration("dmm", "demo")
```

Example - Authorware

```
CallObject(mdb,"dbRegistration","dmm","demo")
```

boolean=dbActiveSQL(sql: string)

The function puts through an SQL command, whose type is SELECT. We are not about to explain the principle of SELECT commands and SQL language. We advise to read appropriate manuals. If the SQL order has been well accomoplished, the function returns true. If an error occured during the SQL order, the function returns false.

Parameters

The only parameter in this function is the SQL command SELECT.

Example - Director

```
global mdb
if mdb.dbActiveSQL("SELECT * FROM data WHERE ordNumber='abc' ") then

end if
```

Example - Authorware

```
CallObject(mdb,"dbActiveSQL","SELECT * FROM data WHERE ordNumber='abc' ")
```

void=dbAppend(record: propertyList))

The function adds a record to the database. New values of the database items are defined in the parameter record. The table has to be opened using the functuion dbOpenTable.

Parameters

The parameter record is of Abstract Data Types type [#fieldByName1: value1, #fieldByName2: value2, #fieldByName3: value3, ...]. For example [#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"].

Example - Director

```
global mdb
mdb.dbAppend([#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"])
```

Example - Authorware

```
CallObject(mdb; "dbAppend", [#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"])
```

void=dbBlobBMP2Member(fieldName: string, castNum: integer, memberNum: integer)

The function saves an item from the database to cast member. The item must be defined in the database as "object OLE" and its format must be BMP. The function only works in Director.

Parameters

The function has 3 parametres. The first, FieldName whose type is string and to which we insert name of the "object OLE" item defined in the database. Cast Num is of integer type and we insert here number of the cast we are going to work with. For example cast internal has number 1 etc. MemberNum is number of the cast member to which we want to save contents of the database. If such a cast member does not exist, it is created: If it does, it is rewritten.

Example - Director

```
global mdb
mdb.dbBlobBMP2Member("field_img",1,2)
```

boolean=dbBof()

The function gives "true" in case the actual record is the first one in the database. Otherwise the function gives "false".

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
if not mdb.dbBof() then
    airPlane=mdb.dbFields()
end if
```

Example - Authorware

```
CallObject(mdb,"dbBof")
```

void=dbClearBookmarks()

The functions deletes all created bookmarks.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
mdb.dbClearBookmarks()
```

Example - Authorware

```
CallObject(mdb,"dbClearBookmarks")
```

void=dbClose()

This function closes and finishes work with an mdb file.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
mdb.dbClose()
```

Example - Authorware

```
CallObject(mdb,"dbClose")
```

integer=dbColumnCount()

The function gives number of fields in the database.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
count=mdb.dbColumnCount()
```

Example - Authorware

```
count:=CallObject(mdb; "dbColumnCount")
```

void=dbColumns()

The function gives names of fields in the database. The values are in Abstract Data Types format.
[#field1: name1, #field2: name2, #field3: name3, ...]
For example [#field1:"ordNumber", #field2:"company", #field3:"type"]

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
fieldsName=mdb.dbColumns()
put fieldsName.field1
put fieldsName.field2
put fieldsName.field3
```

Example - Authorware

```
fieldsName:=CallObject(mdb,"dbColumns")
```

void=dbCompactAndRepair(mdbFile: string)

This function corrects and comprimes the mdb file. This file must not be run neither by any other system nor any function dbOpen of the XTRA.

Parameters

The function has one parameter being mdbFile, which is a name of the mdb file that we want to comprime and correct.

Example - Director

```
global mdb
mdb.dbCompactAndRepair('c:\abc.mdb')
```

Example - Authorware

```
CallObject(mdb; "dbCompactAndRepair","c:\abc.mdb")
```

void=dbDelete()

The function cancels the actual record in the database. The table has to be opened using the functuion dbOpenTable.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
mdb.dbDelete()
```

Example - Authorware

```
CallObject(mdb; "dbDelete")
```

void=dbDeleteBookmark(name: string)

The function deletes a bookmark of the given name.

Parameters

The parameter of the function is name, which represents name of the bookmark.

Example - Director

```
global mdb
mdb.dbDeleteBookmark("test1")
```

Example - Authorware

```
CallObject(mdb; "dbDeleteBookmark","test1")
```

void=dbEdit(record: propertyList)

The function records values of items defined in the record parameter to the actual record. The table has to be opened using the functuion dbOpenTable.

Parameters

The parameter record is of Abstract Data Types type [#fieldByName1: value1, #fieldByName2: value2, #fieldByName3: value3, ...]. For example [#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"].

Example - Director

```
global mdb
mdb.dbEdit([#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"])
```

Example - Authorware

```
CallObject(mdb; "dbEdit", [#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"])
```

boolean=dbEof()

The function gives "true" in case the actual record is the last one in the database. Otherwise the function gives "false" .

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
if not mdb.dbEof() then
    airPlane=mdb.dbFields()
end if
```

Example - Authorware

```
CallObject(mdb,"dbEof")
```

void=dbErrorDialog(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 dbErrorMsg().

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 mdb
mdb.dbErrorDialog(true)
```

Example - Authorware

```
CallObject(mdb,"dbErrorDialog", true)
```

void=dbErrorLog(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 mdb
mdb.dbErrorLog(the pathName&"log.txt", true)
```

Example - Authorware

```
CallObject(mdb; dbErrorLog, FileLocation ^ "log.txt", true)
```

string=dbErrorMsg()

The function gives text of the error that occured when working with the database. If the operation ran correctly, it gives an empty chain.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
mdb.dbOpen(the pathName&"data.mdb", "abc")
error=mdb.dbErrorMsg()
if error="" then airPlane=mdb.dbFields()
```

Example - Authorware

```
error:=CallObject(mdb,"dbErrorMsg")
```

boolean=dbExecSQL(sql: string)

The function puts through an SQL command, whose type is INSERT, UPDATE or DELETE. We are not about to explain the principle of INSERT, UPDATE or DELETE commands and SQL language. We advise to read appropriate manuals. If the SQL order has been well accomoplished, the function returns true. If an error occured during the SQL order, the function returns false.

Parameters

The only parameter in this function is the SQL command INSERT, UPDATE or DELETE.

Example - Director

```
global mdb
if mdb.dbExecSQL("INSERT INTO data (ordNumber, company) VALUES ('dmmMBD27', 'Studio dmm') ") then

end if

if mdb.dbExecSQL("UPDATE data SET registration='YU-AKO'  WHERE ordNumber='A001' ") then

end if

if mdb.dbExecSQL("DELETE FROM data") then

end if
```

Example - Authorware

```
CallObject(mdb; "dbExecSQL","UPDATE data SET registration='YU-AKO'  WHERE ordNumber='A001' ")
```

void=dbFilter(filter: string)

This function enables to set the filter for a certain database table. Using the filter only those records that are in accordance with the setted filter will be screened. It is actually a sort of the "where" clauses in the function select for the question SQL.

Parameters

The only parameter of the function is filter, whose type is string. Filter is a combination of the items, cllauses, operators etc. names, for example (State <> 'CA') AND (Custno > 1400) AND (Custno < 1500). If we use items of the string type in the filter we have to close the searched chain by single quotes. In some cases we have to close separate parts of the filter by brackets, so that the clause could be interpreted correctly.

List of operators that can be used in the function dbFilter:
Operator Meaning
< Less than
> Greater than
>= Greater than or equal to
<= Less than or equal to
= Equal to
<> Not equal to
AND Tests two statements are both True
NOT Tests that the following statement is not True
OR Tests that at least one of two statements is True

Example - Director

```
global mdb
mdb.dbFilter("ordNumber>='A001' and ordNumber<='A027'")
```

Example - Authorware

```
CallObject(mdb; "dbFilter","ordNumber>='A001' and ordNumber<='A027'")
```

void=dbFiltered(filtered: boolean)

This function defines if the filter in the function dbFilter is switched off or switched on.

Parameters

Filtered is the only parameter of the function. For filtered=true the filter is switched on and for filtered=false it is switched off.

Example - Director

```
global mdb
mdb.dbFiltered(true)
```

Example - Authorware

```
CallObject(mdb,"dbFiltered", true)
```

void=dbFirst()

The function sets the first record of the database as the actual one.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
mdb.dbFirst()
```

Example - Authorware

```
CallObject(mdb,"dbFirst")
```

string=dbGetADOVersion()

The function returns number of version of the database engine for work with database files Microsoft Access (mdb), that is installed on the computer. Number of version is returned in the string format.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
version=mdb.dbGetADOVersion()
```

Example - Authorware
```
version:=CallObject(mdb,"dbGetADOVersion")
```

void=dbGetBookmarks()
The function gives a list of defined bookmarks in the Abstract Data Types format ["name1", "name2", "name3", ...]

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
listBookmarks=mdb.dbGetBookmarks()
```

Example - Authorware
```
listBookmarks:=CallObject(mdb,"dbGetBookmarks")
```

value=dbGetFieldByName(fieldName: string)
The function gives value of one item of the actual record in the database. The item is defined by its name. The function gives value in the format in which the item in the database is defined. If, for example the item ordNumber is defined as integer, the given value is integer too. When an eror occurs in database, the function gives back an empty text item.

Parameters
The parameter of the function is fieldName, whose type is string and we insert name of the item defined in the database.

Example - Director
```
global mdb
ordNumber=mdb.dbGetFieldByName("ordNumber")
company=mdb.dbGetFieldByName("company")
type=mdb.dbGetFieldByName("type")
```

Example - Authorware
```
ordNumber:=CallObject(mdb,"dbGetFieldByName","ordNumber")
company:=CallObject(mdb,"dbGetFieldByName","company")
type:=CallObject(mdb,"dbGetFieldByName","type")
```

value=dbGetFieldByNumber(fieldNo: string)
The function gives value of one item of the actual record in the database. The item is defined by its number in the database. The function gives value in the format in which the item in the database is defined. If, for example the item ordNumber is defined as integer, the given value is integer too. When an eror occurs in database, the function gives back an empty text item.

Parameters
The parameter of the function is fieldNo, whose type is integer and we insert number of the item in the database. The parameter ranges from 1 (the first item) to the number dbColumnCount (the last item).

Example - Director
```
global mdb
ordNumber=mdb.dbGetFieldByNumber(1)
company=mdb.dbGetFieldByNumber(2)
type=mdb.dbGetFieldByNumber(3)
```

Example - Authorware

```
ordNumber:=CallObject(mdb,"dbGetFieldByNumber", 1)
company:=CallObject(mdb,"dbGetFieldByNumber", 2)
type:=CallObject(mdb,"dbGetFieldByNumber", 3)
```

list=dbGetFields()

The function gives values of the actual record in the database. The values are in Abstract Data Types format.
[#fieldByName1: value1, #fieldByName2: value2, #fieldByName3: value3, ...]
For example [#ordNumber:"A227", #company:"PAMIR AIR", #type:"B-707-324C"]

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
airPlane=mdb.dbGetFields()
put airPlane.ordNumber
put airPlane.company
put airPlane.type
```

Example - Authorware

```
airPlane:=CallObject(mdb,"dbGetFields")
```

string=dbGetProducer()

The function gives back the text that was placed in the pattern plus the values of each items of the database, that was inserted in the pattern in the <#fieldName> shape. If we place the final text to text cast member, the format html or rtf must be in accordance with the format that is used in Macromedia Director or Macromedia Authorware.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
member("temp").text=mdb.dbGetProducer()
or
member("temp").html=mdb.dbGetProducer()
or
member("temp").rtf=mdb.dbGetProducer()
```

string=dbGetTableNames()

The function returns names of tables and requests, that are included in the Microsoft Access (mdb) file. The function returns the values in the Abstract Data Types format ["nameTable1", "nameTable2", "nameTable3", ...]

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
tables=mdb.dbGetTableNames()
```

Example - Authorware

```
tables:=CallObject(mdb,"dbGetTableNames")
```

void=dbGoToBookmark(name: string)

The function sets as actual record the record that is represented by the given bookmark.

Parameters

The parameter of the function is name, which represents name of the bookmark.

Example - Director

```
global mdb
mdb.dbGoToBookmark("test1")
```

Example - Authorware

```
CallObject(mdb,"dbGoToBookmark","test1")
```

void=dbIndexFieldNames(index: Lists)

The function sets indexes for the table. The table must be opened using the function dbOpenTable. Depending on the setted indexes the table will be put into order as well.

Parameters

The index parameter is in the Abstract Data Types format and it consists of names of the items, that we want to sort the database and index with ["fieldByName1", "fieldByName2", "fieldByName3"...]. For example ["id", "name"].

Example - Director

```
global mdb
mdb.dbIndexFieldNames( ["id", "name"])

end if
```

Example - Authorware

```
CallObject(mdb,"dbIndexFieldNames", ["id"; "name"])
```

void=dbInsertBookmark(name: string)

The function creates and sets bookmark for the actual record of the database. In an application we can create a list of such bookmarks and then visit records that seemed somehow interesting to us.

Parameters

The parameter of the function is name, which represents name of the bookmark. This name must be unique. If we insert an already existing name, the exisiting bookmark will be rewritten.

Example - Director

```
global mdb
mdb.dbInsertBookmark("test1")
```

Example - Authorware

```
CallObject(mdb,"dbInsertBookmark","test1")
```

boolean=dbIsFiltered()

The function finds out if the filter for the database table was switched on using the function dbFiltered(true). If so the function dbIsFiltered returns value true. When using the function dbFiltered(false) value false is returned.

Parameters

There are no parametres in this function.

Example - Director

```
global mdb
if not mdb.dbIsFiltered() then

end if
```

Example - Authorware

```
IsFiltered:=CallObject(mdb,"dbIsFiltered")
```

boolean=dbIsNoError()

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 mdb
mdb.dbOpen(the pathName&"data.mdb", "abc")
if mdb.dbIsNoError() then airPlane=mdb.dbFields()
```

Example - Authorware
```
error:=CallObject(mdb,"dbIsNoError")
```

void=dbLast()
The function sets the last record of the database as the actual one.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
mdb.dbLast()
```

Example - Authorware
```
CallObject(mdb,"dbLast")
```

boolean=dbLocate(listSearch: propertyLists, CaseInSensitive: boolean, PartialKey: boolean)
The function finds in the database a record fulfiling the conditions inserted in the parameter listSearch. If a record fulfiling the conditions is found, the function jumps and returns the value true. In the opposite case the value false is returned.

Parameters
The function has 3 parametres. The listSearch parameter is of the Abstract Data Types type and it consits of a couple of dates being name and value of the item that we want to find [#fieldByName1: value1, #fieldByName2: value2, #fieldByName3: value3, ...]. For example [#id: 10, #name: "John"].
The parameters CaseInSensitive and PartialKey control the way of searching.
If you set CaseInSensitive=true,the searching will be done, no matter what the size of letters is. If you set PartialKey=true the records fulfiling the inserted conditions and parts of their text chain will be found.

Example - Director
```
global mdb
if not mdb.dbLocate([#id: 10, #name: "John"], true, false) then

end if
```

Example - Authorware
```
CallObject(mdb,"dbLocate", [#id: 10, #name: "John"], true, false)
```

void=dbMember2BlobBMP(fieldName: string, castNum: integer, memberNum: integer)
The function saves image cast member to database. In the database the item "object OLE" must be created. The cast member cannot be saved to another type of items. It can only be saved to an existing data sentence. The funciton dbMember2BlobBMP does not insert new records, it only edits the existing ones. The cast member is saved in the database in format BMP. The function only works in Director.

Parameters
The function has 3 parametres. The first, FieldName whose type is string and to which we insert name of the "object OLE" item defined in the database. Cast Num is of integer type and we insert here number of the cast we are going to work with. For example cast internal has number 1 etc. MemberNum is number of the cast member whose contents we want to save to the database. Type of the cast member must be image.

Example - Director
```
global mdb
mdb.dbMember2BlobBMP("field_img",1,2)
```

void=dbMoveBy(distance:integer)
The function jumps n records ahead in the database.

Parameters
Parameter of this function is distance. Distance equals how many records in the database we want to jump. If distance is >0 we go ahead, if it is <0 we go back.

Example - Director
```
global mdb
mdb.dbMoveBy(17)
mdb.dbMoveBy(-2)
```

Example - Authorware
```
CallObject(mdb,"dbMoveBy"; 22)
```

void=dbNext()
The function sets the next record of the database as the actual one.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
mdb.dbNext()
```

Example - Authorware
```
CallObject(mdb,"dbNext")
```

void=dbOpen(mdbFile, mdbPassword: string)
This function opens the mdb file for additional use. If the file is not opened, we cannot work with it.

Parameters
There are 2 parametres in this function. mdbFile is name of mdb file, and mdbPassword is password for the mdb file. If the password was not use when creating the mdb file, we insert an empty chain.

Example - Director
```
global mdb
mdb.dbOpen(the pathName&"data.mdb", "abc")
```

Example - Authorware
```
CallObject(mdb; "dbOpen", FileLocation ^ "data.mdb","abc")
```

void=dbOpenTable(table: string)
The function opens table or request, that are included in the database file mdb. The function dbOpenTable is an alternative to the function dbActiveSQL. To acces the dates both, dbOpenTable and dbActiveSQL can be used.

Parameters
The function has one parameter, which is name of the table or request,that we want to open.

Example - Director
```
global mdb
mdb.dbOpenTable( "data")
```

Example - Authorware
```
CallObject(mdb,"dbOpenTable","data")
```

void=dbPrior()
The function sets theprevious record of the database as the actual one.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
mdb.dbPrior()
```

Example - Authorware
```
CallObject(mdb,"dbPrior")
```

integer=dbRecordCount()
The function gives number of records in the database.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
count=mdb.dbRecordCount()
```

Example - Authorware
```
count:=CallObject(mdb,"dbRecordCount")
```

integer=dbRecordNo()
The function gives record number of the actual record in the database.

Parameters
There are no parametres in this function.

Example - Director
```
global mdb
number=mdb.dbRecordNo()
```

Example - Authorware
```
number:=CallObject(mdb,"dbRecordNo")
```

void=dbSetProducer(template: string, type: symbol)
The function is to define the pattern, which will be used to set the database output. The pattern can be defined in the external file or it can be inserted as a text chain.
ŠThe pattern can contain simple text, which can be in the html, rtf or any other format, for example XML, or even the Abstract Data Types format. The items in the database are defined as texts in the shape <#fieldName>, for example <#id>. If we don't insert the identificator <#fieldName> correctly, the filled values from the database will not be inserted in the pattern.
If we use the pattern in the rtf format or html the whole sign <#fieldName> must be marked in one colour only and it mustn't be divided by any formatting signs. The pattern can be used in any type of the database items, but the items of the memo type.

Parameters
The function has 2 parametres. The parameter teplate is of the string type and we either insert here name of the file containing the appropriate pattern, or the text of the pattern. The way the text in the template parameter will be understood depends on the second parameter that is called type and its type is symbol. It can be of the #file or #string value.

Example - Director

```
global mdb
mdb.dbSetProducer("c:\temp27.txt", #file)
or
mdb.dbSetProducer("c:\temp12.html", #file)
or
mdb.dbSetProducer(member("temp").text, #string)
or
mdb.dbSetProducer(member("temp").html, #string)
or
mdb.dbSetProducer(member("temp").rtf, #string)
```

list=dbXtraVersion()

The function returns information about the XTRA dmmMDB. The function returns values in the format Abstract Data Types:

```
[#fileType: "Xtra (32)",
 #CompanyName: "Studio dmm",
 #FileDescription: "XTRA dmmMDB for work with database files Microsoft Access",
 #FileVersion: "1.9.0.21", #InternalName: "dmmMDB",
 #LegalCopyRight: "© 1992-2005 Studio dmm",
 #LegalTradeMarks: "Studio dmm",
 #OriginalFileName: "dmmMDB.x32",
 #productName: "dmmMDB",
 #productVersion: "1.9.0.0"]
```

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 mdb
version=mdb.dbXtraVersion()
```

Example - Authorware

```
version:=CallObject(mdb,"dbXtraVersion")
```