GAMS Data Exchange API

System and Reference Manual, created on 7/30/2014.

Copyright (c) 2002-2012 GAMS Development Corporation. All rights reserved.

Content

1 GDX GAMS Data Exchange

This document describes the Application Programmers Interface (API) for the GDX library. The GDX library is used to read or write GDX files. A GDX file is a file that stores the values of one or more GAMS symbols such as sets, parameters variables and equations. GDX files can be used to prepare data for a GAMS model, present results of a GAMS model, store results of the same model using different parameters etc. A GDX file does not store a model formulation or executable statements.

GDX files are binary files that are portable between different platforms. They are written using the byte ordering native to the hardware platform they are created on, but can be read on a platform using a different byte ordering.

To read or write data, we need to be able to reference the set elements used to represent the index space for symbols with one or more dimensions. The API provides three interface models for this purpose:

1. The **String** based interface. An n dimensional element is represented as an array of strings.
2. The **Raw** integer interface. An n dimensional element is represented as an array of integers. The integer used for each index position is obtained from the API after registering the string representation with the API.
3. The **Mapped** integer interface. An n dimensional element is represented as an array of integers. The integer used for each index position is defined by the user. Before such an element can be used, its value and string has to be registered.

Moving code used with GAMS 22.5 needs some editing to support the new feautures available in version 22.6; see Conversion Issues (see Conversion issues when moving from GAMS 22.5 to 22.6, page 29).

Next: Writing Data (see Writing data to a GDX file, page 1) or Reading Data (see Reading data from a GDX file, page 3)

1.1 Writing data to a GDX file

Creating a GDX file and writing one or more symbols to the file requires a number of steps:

1. Make sure the GDX library is available
2. Open a file for writing
3. Register unique elements
4. Start writing a symbol
5. Write the data
6. Finish writing for the symbol
7. Optional: share acronyms
8. Close the file
9. Unload the GDX library

Steps 3 - 6 can be repeated to write any number of symbols to the file. Once a symbol has been written to the file, it cannot be replaced. Currently, there are no facilities to overwrite a symbol or append data to an existing file.

The following sections illustrate the basic steps for each type of interface. The method of writing (string, raw or mapped) can be selected for each symbol; it cannot be changed while writing a symbol.

Next: Write Using Strings (see Writing data using strings, page 1) or Write Using Integers (see Writing data using integers (Raw), page 2) or Write Using User Defined Integers (see Writing data using integers (Mapped), page 2)

1.1.1 Writing data using strings

The String based interface is suitable when we want to use a string based index and do not want to maintain a mapping from strings to integers.

Before writing data using a string based interface we can register strings for the unique elements, but this step is optional. The only reason to register the strings beforehand is to enter the strings in a given order which may have advantages later in the modelling stage.

```
if not gdxDataWriteStrStart(PGX,'Demand','Demand data',1,Ord(dt_par),0)
then
   ReportGDXError(PGX);

IndxS[1] := 'New-York';
Values[1] := 324.0;
gdxDataWriteStr(PGX,IndxS,Values);

IndxS[1] := 'Chicago';
Values[1] := 299.0;
gdxDataWriteStr(PGX,IndxS,Values);

if not gdxDataWriteDone(PGX)
then
    ReportGDXError(PGX);
```

In this example we write two records for a parameter that has a dimension of one.

1.1.2 Writing data using integers (Raw)

The Raw interface is suitable when we want to manage our own list of unique elements, and use an integer based index. The Raw interface assumes that the integers assigned to the strings range from one to the number of strings registered.

Before we can write data using the Raw interface, we have to register the strings for the unique elements. The GDX routines will assign an integer to the string that increases by one for every string registered.

```
if not gdxUELRegisterRawStart(PGX)
then
   ReportGDXError(PGX);
gdxUELRegisterRaw(PGX,'New-York');
gdxUELRegisterRaw(PGX,'Chicago');
if not gdxUELRegisterDone(PGX)
then
   ReportGDXError(PGX);

if not gdxDataWriteRawStart(PGX,'Demand','Demand data',1,Ord(dt_par),0)
then
   ReportGDXError(PGX);

IndxI[1] := 1;
Values[1] := 324.0;
gdxDataWriteRaw(PGX,IndxI,Values);

IndxI[1] := 2;
Values[1] := 299.0;
gdxDataWriteRaw(PGX,IndxS,Values);

if not gdxDataWriteDone(PGX)
then
    ReportGDXError(PGX);
```

1.1.3 Writing data using integers (Mapped)

The Mapped interface is suitable when we want to manage our own list of unique elements, and use an integer based index. The mapped interface lets us select our own mapping between strings for the unique elements and their integer equivalent. The integers assigned to the unique elements should be greater equal one, and be unique for each element.

Before we can write data using the Mapped interface, we have to register the strings for the unique elements.

```
if not gdxUELRegisterMapStart(PGX)
then
   ReportGDXError(PGX);
gdxUELRegisterMap(PGX,1000,'New-York');
gdxUELRegisterMap(PGX,2000,'Chicago');
if not gdxUELRegisterDone(PGX)
then
   ReportGDXError(PGX);

if not gdxDataWriteMapStart(PGX,'Demand','Demand data',1,Ord(dt_par),0)
then
   ReportGDXError(PGX);

IndxI[1] := 1000;
Values[1] := 324.0;
gdxDataWriteRaw(PGX,IndxI,Values);

IndxI[1] := 2000;
Values[1] := 299.0;
gdxDataWriteRaw(PGX,IndxS,Values);

if not gdxDataWriteDone(PGX)
then
    ReportGDXError(PGX);
```

In this example we register two unique elements, and write a parameter of dimension one.

1.2 Reading data from a GDX file

Opening an existing GDX file and reading one or more symbols from the file requires a number of steps:

1. Make sure the GDX library is available
2. Open a file for reading
3. Optional: share acronyms
4. Register unique elements
5. Start reading a symbol
6. Read the data
7. Finish reading for the symbol
8. Close the file
9. Unload the GDX library

Steps 3 - 6 can be repeated to read any number of symbols from the file.

The following sections illustrate the basic steps for each type of interface. The method of writing (string, raw or mapped) can be selected for each symbol; it cannot be changed while writing a symbol.

Next: Read Using Strings (see Reading data using strings, page 3) or Read Using Integers (see Reading data using integers (Raw), page 4) or Read Using User Defined Integers (see Reading data using integers (Mapped), page 5)

1.2.1 Reading data using strings

Reading data using strings does not require any unique element registration.

```
if not gdxFindSymbol(PGX,'x',SyNr)
```

```
then
   begin
   WriteLn('**** Could not find symbol X');
   halt;
   end;

gdxSymbolInfo(PGX,SyNr,SyName,SyDim,SyTyp);
if (SyDim <> 1) or (SyTyp <> Ord(dt_par))
then
   begin
   WriteLn('**** X is not a one dimensional parameter');
   halt;
   end;

if not gdxDataReadStrStart(PGX,SyNr,NrRecs)
then
   ReportGDXError(PGX);

WriteLn('Parameter X has ',NrRecs,' records');

while gdxDataReadStr(PGX,IndxS,Values,N)
do WriteLn('Record = ',IndxS[1],' ',Values[1]);

if not gdxDataReadDone(PGX)
then
   ReportGDXError(PGX);
```

In this example we find the symbol by its name, and before reading the data we verify that the symbol represents a one dimensional parameter.

1.2.2 Reading data using integers (Raw)

Reading data using integers in Raw mode does not require the registration of unique elements. The read routine returns an integer for which we can find the string representation.

```
if not gdxFindSymbol(PGX,'x',SyNr)
then
   begin
   WriteLn('**** Could not find symbol X');
   halt;
   end;

gdxSymbolInfo(PGX,SyNr,SyName,SyDim,SyTyp);
if (SyDim <> 1) or (SyTyp <> Ord(dt_par))
then
   begin
   WriteLn('**** X is not a one dimensional parameter');
   halt;
   end;

if not gdxDataReadRawStart(PGX,SyNr,NrRecs)
then
   ReportGDXError(PGX);

WriteLn('Parameter X has ',NrRecs,' records');

while gdxDataReadRaw(PGX,IndxI,Values,N)
do begin
   Write('Record = ',IndxI[1],' = ',Values[1]);
   gdxUMUelGet(PGX,IndxI[1],S,UsrMap);
   WriteLn(' with string = ',S);
   end;

if not gdxDataReadDone(PGX)
then
   ReportGDXError(PGX);
```

In this example we find the symbol by its name, and before reading the data we verify that the symbol represents a one dimensional parameter. When reading the data, we get a unique element as an integer. The integer value is used to get the corresponding string for the unique element.

1.2.3 Reading data using integers (Mapped)

Reading data using integers in Mapped mode requires the registration of unique elements. The read routine returns an integer for which we can find the string representation.

When the gdx file contains data elements that we never registered, the read function will not return these elements, they will be added to an internal list of error records instead. The next topic, Reading data using a filter (see page 6) shows a more detailed example.

```
if not gdxUELRegisterMapStart(PGX)
then
   ReportGDXError(PGX);
gdxUELRegisterMap(PGX,1000,'New-York');
gdxUELRegisterMap(PGX,2000,'Chicago');
if not gdxUELRegisterDone(PGX)
then
   ReportGDXError(PGX);

if not gdxFindSymbol(PGX,'x',SyNr)
then
   begin
   WriteLn('**** Could not find symbol X');
   halt;
   end;

gdxSymbolInfo(PGX,SyNr,SyName,SyDim,SyTyp);
if (SyDim <> 1) or (SyTyp <> Ord(dt_par))
then
   begin
   WriteLn('**** X is not a one dimensional parameter');
   halt;
   end;

if not gdxDataReadMapStart(PGX,SyNr,NrRecs)
then
   ReportGDXError(PGX);

WriteLn('Parameter X has ',NrRecs,' records');

for N := 1 to NrRecs
do begin
   if gdxDataReadMap(PGX,N,IndxI,Values,N)
   then
      begin
      Write('Record = ',N,'  ',IndxI[1],' = ',Values[1]);
      GetUEL(PGX,IndxI[1],S);
      WriteLn(' with string = ',S);
      end;
   end;

if not gdxDataReadDone(PGX)
then
   ReportGDXError(PGX);

NrRecs := gdxDataErrorCount(PGX);
if NrRecs > 0
then
   WriteLn(NrRecs,' records were skipped');
```

In this example we register a few unique elements using our own integer values. After verifying that we can find the symbol and that the symbol represents a one dimensional parameter we can read the data. The index for the parameter is returned using the integers we used when registering our elements. When we read the records in sequence, the index returned will be sorted with the first index position the most significant.

After reading the data, we print the number of records that were skipped in the read routine.

.

1.2.4 Reading data using a filter

Reading data using a filter allows us to control the action for every index position. The type of action is specified using action codes and needs to be specified for every index position. The actual reading of the records is done with the gdxDataReadMap (see page 80) function.

Action code	
UnMapped (-2)	No mapping is performed; the value of the unique element is the value as stored in the GDX file. Use gdxUMUelGet (see page 97) to get the string representation.
Checked (0)	Map the unique element value to the user defined value. Use gdxGetUEL (see page 89) to get the string representation. If a user mapping was not defined for this element , the record is flagged as an error record and the record will be skipped.
Expand (-1)	Map the unique element value to the user defined value. Use gdxGetUEL (see page 89) to get the string representation. If a user mapping was not defined for this element, define a user mapping automatically using the next higher user map value.
Filter Number (>0)	Map the unique element value to the user defined value. Use gdxGetUEL (see page 89) to get the string representation. If the element is not enabled in the filter for this index position, the record is flagged as an error record and it will be skipped. The filter number is specified using the gdxFilterRegisterStart (see page 86) function.

Refering to the following GAMS fragment, we want to read the parameter A. The set I is the domain for the first index; there is no domain for the second index position:

```
Set I /.../;
Parameter A(I,*);
```

Assuming we have read set I already, the following code snapshot illustrates how to read parameter A.

```
// Register the filter for set I; reference this filter with integer 123
if not gdxFilterRegisterStart(PGX,123)
then
   ReportGDXError(PGX);
gdxFilterRegister(PGX,1000);
gdxFilterRegister(PGX,2000);
if not gdxFilterRegisterDone(PGX)
then
   ReportGDXError(PGX);

// set the filter
Filt[1] := 123; //filter for I
Filt[2] := -1; // expand

// Remember highest mapped value in variable LastMapped
gdxUMUelInfo(PGX,NrUnMapped,LastMapped);

// Read parameter A as a 2 dimensional parameter
if not gdxFindSymbol(PGX,'A',SyNr)
then
   begin
   WriteLn('**** Could not find symbol A');
   halt;
   end;
```

```
gdxSymbolInfo(PGX,SyNr,SyName,SyDim,SyTyp);
if (SyDim <> 2) or (SyTyp <> Ord(dt_par))
then
   begin
   WriteLn('**** A is not a two dimensional parameter');
   halt;
   end;

if not gdxReadFilteredStart(PGX,SyNr,Filt,NrRecs);
then
   ReportGDXError(PGX);

for N := 1 to NrRecs
do begin
   if gdxDataReadMap(PGX,N,IndxI,Values)
   then
      begin
      //do something with the record read
      end;
   end;
if not gdxDataReadDone(PGX)
then
   ReportGDXError(PGX);

// see if there are new unique elements
gdxUMUelInfo(PGX,NrUnMapped,NewLastMapped);
if NewLastMapped > LastMapped
then
   begin
   for N := LastMapped + 1 to NewLastMapped
   do begin
      gdxGetUel(PGX,N,S);
      WriteLn('New element ',N,' = ',S);
      end;
   end;
```

1.3 Dealing with acronyms

In GAMS we can use acronyms in places where we can use a floating point number as in the following example:

```
set i /i1*i5/;
acronym acro1, acro2;
parameter A(i) /i1=1, i2=acro1, i3=3, i4=acro2, i5=5/;
display A;
```

The result of the display statement looks like:

```
----      4 PARAMETER A

i1 1.000,    i2 acro1,    i3 3.000,    i4 acro2,    i5 5.000
```

As we write data to a GDX file, the system keeps track which acronyms were used in the data written. Before we close the GDX file, we share the identifiers used for each acronym used. When reading a GDX file, we share all acronym identifiers and their corresponding index before reading any data. Doing so will replace the acronym indices stored in the GDX file by the one we provide.

The example below illustrates these steps.

```
program acronyms;

{$APPTYPE CONSOLE}

uses
  sysutils,
  gxdefs,
  gmsspecs,
  gdxdpdef;

var
   PGX     : PGXFile;
   NrRecs  : integer;
```

```
    UELS    : TgdxUELIndex;
    Vals    : TgdxValues;
    FDim    : integer;
    N       : integer;
    ErrMsg  : shortstring;
    ErrNr   : integer;
    acrname : shortstring;
    acrtext : shortstring;
    acrindx : integer;
 begin
 //Check the library
 if not gdxGetReadyX(ErrMsg)
 then
    begin
    WriteLn('Error loading GDX library, msg = ', ErrMsg);
    Halt(1);
    end;
 //Create GDX object and open file for writing
 gdxCreateX(PGX, ErrMsg);
 gdxOpenWriteEx(PGX, 'test.gdx', 'testing', 0, ErrNr);

 //register some unique elements
 gdxUELRegisterRawStart(PGX);
 for N := 1 to 5
 do gdxUELRegisterRaw(PGX, 'uel' + IntToStr(N));
 gdxUELRegisterDone(PGX);

 //write a parameter with two acronyms
 gdxDataWriteRawStart(PGX, 'symb1', 'text for symb1', 1, Ord(dt_par), 0);
 for N := 1 to 5
 do begin
    UELS[1] := N;
    if N in [2, 4]
    then
       Vals[vallevel] := gdxAcronymValue(PGX, N)
    else
       Vals[vallevel] := N;
    gdxDataWriteRaw(PGX, UELS, Vals);
    end;
 gdxDataWriteDone(PGX);

 //provide the names for the acronyms used
 for N := 1 to gdxAcronymCount(PGX)
 do begin
    gdxAcronymGetInfo(PGX, N, acrname, acrtext, acrindx);
    if acrindx = 2
    then
       gdxAcronymSetInfo(PGX, N, 'acro1', 'Some text for acro1', acrindx)
    else
       if acrindx = 4
       then
          gdxAcronymSetInfo(PGX, N, 'acro2', 'Some text for acro2', acrindx)
    end;

 //final check for errors before we close the file
 N := gdxClose(PGX);
 if N <> 0
 then
    begin
    gdxErrorStr(Nil, N, ErrMsg);
    WriteLn('Error writing file = ', ErrMsg);
    Halt(1);
    end;
 gdxFree(PGX);

 //open the file we just created
 gdxCreateX(PGX, ErrMsg);
 gdxOpenRead(PGX, 'test.gdx', ErrNr);
 if ErrNr <> 0
 then
```

```
      begin
      WriteLn('Error opening file, nr = ', ErrNr);
      Halt(1);
      end;

   //give acronym indices using the name of the acronym
   gdxAcronymSetInfo(PGX, 1, 'acro1', '', 1000);
   gdxAcronymSetInfo(PGX, 2, 'acro2', '', 1001);

   //read the parameter
   gdxDataReadRawStart(PGX, 1, NrRecs);
   while gdxDataReadRaw(PGX, UELs, Vals, FDim) <> 0
   do begin
      N := gdxAcronymIndex(PGX, Vals[vallevel]);
      if N = 0
      then
         WriteLn(Vals[vallevel])
      else
         WriteLn('Acronym: index = ', N)
      end;
   gdxDataReadDone(PGX);

   ErrNr := gdxClose(PGX);
   //final error check before closing the file
   if ErrNr <> 0
   then
      begin
      gdxErrorStr(nil, ErrNr, ErrMsg);
      WriteLn('Error reading file = ', ErrMsg);
      Halt(1);
      end;
   gdxFree(PGX);
   end.
```

1.4 Functions by Category

The following table organizes the functions by category:

File Open/Close	gdxOpenRead (see page 90) gdxOpenWrite (see page 90) gdxClose (see page 77)
System/Symbol Information	gdxSystemInfo (see page 95) gdxSymbolInfo (see page 94) gdxSymbolInfoX (see page 94) gdxFindSymbol (see page 86) gdxGetUEL (see page 89)
Unique elements	gdxUELRegisterRawStart (see page 96) gdxUELRegisterMapStart (see page 96) gdxUELRegisterStrStart (see page 97) gdxUELRegisterRaw (see page 96) gdxUELRegisterMap (see page 96) gdxUELRegisterStr (see page 96) gdxUELRegisterDone (see page 95) gdxGetUEL (see page 89) gdxUMUelInfo (see page 97) gdxUMUelGet (see page 97) gdxUMFindUEL (see page 97)
Write Data	gdxDataWriteRawStart (see page 84) gdxDataWriteMapStart (see page 83) gdxDataWriteStrStart (see page 84) gdxDataWriteRaw (see page 83) gdxDataWriteMap (see page 83) gdxDataWriteStr (see page 84) gdxDataWriteDone (see page 83)
Read Data	gdxDataReadRawStart (see page 81) gdxDataReadMapStart (see page 80) gdxDataReadStrStart (see page 82) gdxDataReadRaw (see page 81) gdxDataReadMap (see page 80) gdxDataReadStr (see page 82) gdxDataReadFilteredStart (see page 80) gdxDataReadDone (see page 80) gdxDataErrorCount (see page 79) gdxDataErrorRecord (see page 79)
Text for unique elements	gdxAddSetText (see page 77) gdxSetTextNodeNr (see page 92) gdxGetElemText (see page 87) gdxSetHasText (see page 91)
Filters	gdxFilterRegisterStart (see page 86) gdxFilterRegister (see page 85) gdxFilterRegisterDone (see page 86) gdxFilterExists (see page 85)

Special Values	gdxResetSpecialValues (see page 91) gdxSetSpecialValues (see page 92) gdxGetSpecialValues (see page 89) gdxMapValue (see page 90)
Errors	gdxGetLastError (see page 87) gdxErrorCount (see page 84) gdxErrorStr (see page 85)
Version Information	gdxSetTraceLevel (see page 92) gdxFileVersion (see page 85) gdxGetDLLVersion (see page 87)
Longest symbol unique element	gdxSymbMaxLength (see page 93) gdxUELMaxLength (see page 95) gdxSymbIndxMaxLength (see page 92)
Acronyms	gdxAcronymIndex (see page 75) gdxAcronymValue (see page 76) gdxAcronymCount (see page 75) gdxAcronymGetInfo (see page 75) gdxAcronymSetInfo (see page 76)

1.5 Transition diagram

The routines documented below follow certain input / output state transitions. Routines not documented below have no special state requirements.

Routine	Input State	Output State	Notes
gdxOpenRead (see page 90)	f_notopen	fr_init	
gdxOpenWrite (see page 90)	f_notopen	fw_init	
gdxOpenWriteEx (see page 91)	f_notopen	fw_init	
gdxClose (see page 77)	fr_init, fw_init	f_notopen	
gdxDataWriteRawStart (see page 84)	fw_init	fw_raw_data	
gdxDataWriteMapStart (see page 83)	fw_init	fw_map_data	
gdxDataWriteStrStart (see page 84)	fw_init	fw_str_data	
gdxDataWriteRaw (see page 83)	fw_raw_data	N/C	
gdxDataWriteMap (see page 83)	fw_map_data	N/C	
gdxDataWriteStr (see page 84)	fw_str_data	N/C	
gdxDataWriteDone (see page 83)	fw_raw_data, fw_map_data, fw_str_data, fw_init	fw_init	
gdxDataReadRawStart (see page 81)	fr_init	fr_raw_data	Note1
gdxDataReadMapStart (see page 80)	fr_init	fr_map_data	Note1
gdxDataReadStrStart (see page 82)	fr_init	fr_str_data	Note1
gdxDataReadFilteredStart (see page 80)	fr_init	fr_map_data	Note1
gdxDataReadRaw (see page 81)	fr_raw_data	N/C, fr_init	Note2
gdxDataReadMap (see page 80)	fr_map_data	N/C, fr_init	Note2
gdxDataReadStr (see page 82)	fr_str_data	N/C, fr_init	Note2
gdxDataReadDone (see page 80)	fr_raw_data, fr_map_data, fr_str_data, fr_init		

gdxDataErrorRecord (see page 79)	fr_init, fr_map_data, fw_raw_data, fw_map_data, fw_str_data		
gdxFilterRegisterStart (see page 86)	fr_init	fr_filter	
gdxFilterRegister (see page 85)	fr_filter	N/C	
gdxFilterRegisterDone (see page 86)	fr_filter	fr_init	
gdxFilterExists (see page 85)	fr_init	N/C	
gdxUELRegisterRawStart (see page 96)	fr_init	f_raw_elem	
gdxUELRegisterRaw (see page 96)	f_raw_elem	N/C	
gdxUELRegisterMapStart (see page 96)	fr_init	f_map_elem	
gdxUELRegisterMap (see page 96)	f_map_elem	N/C	
gdxUELRegisterStrStart (see page 97)	fr_init	f_str_elem	
gdxUELRegisterStr (see page 96)	f_str_elem	N/C	
gdxUELRegisterDone (see page 95)	f_raw_elem, f_map_elem, f_str_elem	fr_init	
gdxSymbMaxLength (see page 93)	fr_init	N/C	
gdxUELMaxLength (see page 95)	fr_init	N/C	
gdxSymbIndxMaxLength (see page 92)	fr_init	N/C	
gdxAcronymSetInfo (see page 76)	fr_init, fw_init	N/C	

Note1: New state assumes there is data; when the symbol is empty, the state will be fr_init.

Note2: No change in state when there is still data; when we reach the end of the data the new state will be fr_init.

1.6 Example programs

Some complete example programs are illustrated in the following topics.

- GAMS and Delphi (see Example 1, page 11)
- gdxdump in C (see Example 2: C program, page 15)
- program in C++ (see Example 3: C++ program, page 18)
- program in VB.NET (see Example 4: VB.NET program, page 19)
- program in Fortran (see Example 5: Fortran program, page 22)
- program in Python (see Example 6: Python program, page 24)
- program in C# (see Example 7: C# program, page 25)
- program in Java (see Example 8: Java program, page 27)

1.6.1 Example 1

In this modified version of the trnsport.gms model, we use an external program to generate data for the demand parameter. After we solve the model, we write the solution to a GDX file, and call the external program again to read the variable from the GDX file.

The modified trnsport.gms model:

```
$Title trnsport model using gdx files
$EOLCOM //
  Sets
```

```
        i   canning plants   / seattle, san-diego /
        j   markets          / new-york, chicago, topeka / ;

   Parameter
        a(i)  capacity of plant i in cases
          /    seattle     350
               san-diego   600  /

        b(j)  demand at market j in cases ;

   Table d(i,j)  distance in thousands of miles
                     new-york       chicago      topeka
       seattle          2.5           1.7          1.8
       san-diego        2.5           1.8          1.4  ;

   Scalar f  freight in dollars per case per thousand miles  /90/ ;

   Parameter c(i,j)  transport cost in thousands of dollars per case ;

             c(i,j) = f * d(i,j) / 1000 ;

   Variables
        x(i,j)  shipment quantities in cases
        z       total transportation costs in thousands of dollars ;

   Positive Variable x ;

   Equations
        cost        define objective function
        supply(i)   observe supply limit at plant i
        demand(j)   satisfy demand at market j ;

  // These lines execute during the compilation phase
  // The GAMS system directory is passed the the program so it knows where
  // to look for the gdxdclib library

  $call 'gdxexdp.exe %gams.sysdir%'      // create demand data
  $GDXIN demanddata.gdx                  // open data file
  $LOAD  b=demand                        // load parameter b (named 'demand' in file)
  $GDXIN                                 // close data file

   cost ..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

   supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;

   demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

   Model transport /all/ ;

   Solve transport using lp minimizing z ;

   Display b,x.l, x.m ;

  // These lines execute during the execution phase
  execute_unload 'results.gdx',x;                    // write variable x to the gdx file
  execute 'gdxexdp.exe %gams.sysdir% results.gdx'; // do something with the solution
```

The external program is illustrated in Delphi (see Example 1 in Delphi, page 12)

Example 1 in Delphi

Please note that the Delphi program also has been written in VB.NET; see VB.NET Example (see Example 4: VB.NET program, page 19).

```
program xp_example1;

///////////////////////////////////////////////////////////////
// This program generates demand data for a modified version //
// of the trnsport model or reads the solution back from a   //
// gdx file.                                                 //
//                                                           //
// Calling convention:                                           //
// Case 1:                                                       //
//    Parameter 1: GAMS system directory                         //
// The program creates a GDX file with demand data               //
// Case 2:                                                       //
//    Parameter 1: GAMS system directory                         //
//    Parameter 2: gdxfile                                       //
// The program reads the solution from the GDX file              //
// Paul van der Eijk Jun-12, 2002                                //
///////////////////////////////////////////////////////////////////

{$APPTYPE CONSOLE}
{$H- short strings}

uses
  sysutils,
  gxdefs,
  gmsspecs,
  gdxdcpdef;

procedure ReportGDXError(PGX: PGXFile);
var
   S: ShortString;
begin
WriteLn('**** Fatal GDX Error');
GDXErrorStr(nil, GDXGetLastError(PGX),S);
WriteLn('**** ',S);
Halt(1);
end;

procedure ReportIOError(N: integer);
begin
WriteLn('**** Fatal I/O Error = ',N);
Halt(1);
end;

var
   PGX  : PGXFile;

procedure WriteData(const s: string; V: double);
var
   Indx  : TgdxStrIndex;
   Values: TgdxValues;
begin
Indx[1] := s;
Values[vallevel] := V;
GDXDataWriteStr(PGX,Indx,Values);
end;


var
   Msg     : string;
   Sysdir  : string;
   Producer: string;
   ErrNr   : integer;
   Indx    : TgdxStrIndex;
   Values  : TgdxValues;
   VarNr   : integer;
   NrRecs  : integer;
   N       : integer;
   Dim     : integer;
   VarName : shortstring;
   VarTyp  : integer;
   D       : integer;

begin
if not(ParamCount in [1,2])
then
   begin
   WriteLn('**** XP_Example1: incorrect number of parameters');
```

```
    Halt(1);
    end;

  sysdir := ParamStr(1);
  WriteLn('XP_Example1 using GAMS system directory: ',sysdir);

  if not GDXCreateD(PGX,sysdir,Msg)
  then
    begin
    WriteLn('**** Could not load GDX library');
    WriteLn('**** ', Msg);
    exit;
    end;

  GDXGetDLLVersion(nil, Msg);
  WriteLn('Using GDX DLL version: ',Msg);

  if ParamCount = 1
  then
    begin
    //write demand data
    GDXOpenWrite(PGX,'demanddata.gdx','xp_example1', ErrNr);
    if ErrNr <> 0
    then
      ReportIOError(ErrNr);

    if GDXDataWriteStrStart(PGX,'Demand','Demand data',1,gms_dt_par,0) = 0
    then
      ReportGDXError(PGX);
    WriteData('New-York',324.0);
    WriteData('Chicago' ,299.0);
    WriteData('Topeka'  ,274.0);
    if GDXDataWriteDone(PGX) = 0
    then
      ReportGDXError(PGX);

    WriteLn('Demand data written by xp_example1');
    end
  else
    begin
    //read x variable back (non-default level values only)
    GDXOpenRead(PGX,ParamStr(2), ErrNr);
    if ErrNr <> 0
    then
      ReportIOError(ErrNr);

    GDXFileVersion(PGX,Msg,Producer);
    WriteLn('GDX file written using version: ',Msg);
    WriteLn('GDX file written by: ',Producer);

    if GDXFindSymbol(PGX,'x',VarNr) = 0
    then
      begin
      WriteLn('**** Could not find variable X');
      Halt(1);
      end;

    GDXSymbolInfo(PGX,VarNr,VarName,Dim,VarTyp);
    if (Dim <> 2) or (VarTyp <> gms_dt_var)
    then
      begin
      WriteLn('**** X is not a two dimensional variable');
      Halt(1);
      end;

    if GDXDataReadStrStart(PGX,VarNr,NrRecs) = 0
    then
      ReportGDXError(PGX);

    WriteLn('Variable X has ',NrRecs,' records');
```

```
     while GDXDataReadStr(PGX,Indx,Values,N) <> 0
     do begin
        if Values[vallevel] = 0.0       //skip level = 0.0 is default
        then
           continue;
        for D := 1 to Dim
        do begin
           Write(Indx[D]);
           if D < Dim
           then
              Write('.');
           end;
        WriteLn(' = ',Values[vallevel]:7:2);
        end;
     WriteLn('All solution values shown');
     GDXDataReadDone(PGX);
     end;

  ErrNr := GDXClose(PGX);
  if ErrNr <> 0
  then
     ReportIOError(ErrNr);

  end.
```

1.6.2 Example 2: C program

This is a simplified version of the gdxdump program written in C

```
/*
  Use this command to compile the example:
  cl gdxdumpc.c ../../gmstest/apifiles/C/api/gdxcc.c ../../gmstest/apifiles/C/api/gclgms.c -
I../../gmstest/apifiles/C/api/
*/

#include <stdio.h>
#include <string.h>
#include "gdxcc.h"
#include "gclgms.h"

char *val2str(gdxHandle_t Tptr, double val, char *s) {

  int sv;

  if (gdxAcronymName(Tptr, val, s)) {
    return s;
  } else {
    gdxMapValue(Tptr, val, &sv);
    if (sv_normal != sv)
      sprintf(s,"%s", gmsSVText[sv]);
    else
      sprintf(s,"%g", val);
    return s;
  }
}

int main (int argc, char *argv[]) {

  int rc,i,j,NrSy,NrUel,ADim,ACount,AUser,AUser2,NRec,FDim,IDum, BadUels=0;
  int ATyp, ATyp2;
  char
    msg[GMS_SSSIZE],
    FileVersion[GMS_SSSIZE], FileProducer[GMS_SSSIZE],
    sName[GMS_SSSIZE], sName2[GMS_SSSIZE], sText[GMS_SSSIZE], UelName[GMS_SSSIZE];

  gdxHandle_t Tptr=NULL;
  char DomainIDs[GMS_MAX_INDEX_DIM][GMS_SSSIZE];
  char *DP[GMS_MAX_INDEX_DIM];
  double
    Vals[GMS_VAL_MAX],
    dv[GMS_VAL_MAX];
  int
    Keys[GMS_MAX_INDEX_DIM];
   char *dn, c;

  GDXSTRINDEXPTRS_INIT(DomainIDs,DP);

  if (argc != 2) {
    printf("Usage: gdxdumpc gdxfilen");
    exit(1);
  }

  gdxCreate (&Tptr,msg,sizeof(msg));
  if (NULL==Tptr) {
    printf("Could not create GDX object:n%sn",msg);
    exit(1);
  }

  rc = gdxOpenRead(Tptr, argv[1], &i);
  if (0==rc) {
    gdxErrorStr(Tptr,i,msg);
    printf("Could not read GDX file %s:n%s (rc=%d)n",argv[1],msg,rc);
    exit(1);
  }

  rc = gdxGetLastError(Tptr);
  if (rc) {
    gdxErrorStr(Tptr,rc,msg);
    printf("Problems processing GDX file %s:n%s (rc=%d)n",argv[1],msg,rc);
    exit(1);
  }

  gdxFileVersion(Tptr, FileVersion, FileProducer);
  gdxSystemInfo(Tptr,&NrSy,&NrUel);
  printf("*  File version   : %sn",FileVersion);
  printf("*  Producer       : %sn",FileProducer);
  printf("*  Symbols        : %dn",NrSy);
  printf("*  Unique Elements: %dn",NrUel);

 /* Acroynms */
 for (i=1; i<=gdxAcronymCount(Tptr); i++) {
   gdxAcronymGetInfo(Tptr, i, sName, sText, &rc);
   printf("Acronym %s", sName);
   if (strlen(sText)) printf(" '%s'", sText);
   printf(";n");
 }

 /* Symbolinfo */
 printf("$ontextn");
 for (i=1; i<=NrSy; i++) {
   gdxSymbolInfo(Tptr, i, sName, &ADim, &ATyp);
   gdxSymbolInfoX(Tptr, i, &ACount, &rc, sText);
   printf("%-15s %3d %-12s %sn", sName, ADim, gmsGdxTypeText[ATyp],sText);
 }
 printf("$offtextn");

 printf("$onempty onembedded n");
 for (i=1; i<=NrSy; i++) {

   gdxSymbolInfo(Tptr, i, sName, &ADim, &ATyp);
   gdxSymbolInfoX(Tptr, i, &ACount, &AUser, sText);

   if (GMS_DT_VAR == ATyp || GMS_DT_EQU == ATyp) printf("$ontextn");

   if (GMS_DT_VAR == ATyp) {
     if (AUser < 0 || AUser>=GMS_VARTYPE_MAX) AUser = GMS_VARTYPE_FREE;
     memcpy(dv,gmsDefRecVar[AUser],GMS_VAL_MAX*sizeof(double));
     dn = (char *) gmsVarTypeText[AUser];
   } else if (GMS_DT_EQU == ATyp) {
     if (AUser < 0 || AUser>=GMS_EQUTYPE_MAX) AUser = GMS_EQUTYPE_E;
     memcpy(dv,gmsDefRecEqu[AUser],GMS_VAL_MAX*sizeof(double));
   } else dv[GMS_VAL_LEVEL] = 0.0;


   if (0 == ADim && GMS_DT_PAR == ATyp) /* Scalar */
     printf("Scalar");
   else {
     if (GMS_DT_VAR == ATyp) printf("%s ",dn);
     printf("%s",gmsGdxTypeText[ATyp]);
   }
   if (GMS_DT_ALIAS == ATyp) {
     gdxSymbolInfo(Tptr, AUser, sName2, &j, &ATyp2);
     printf(" (%s, %s);n", sName, sName2);
   } else {
     printf(" %s", sName);
     if (ADim > 0) {
       gdxSymbolGetDomain(Tptr, i, Keys);
       printf("("); for (j=0; j<ADim; j++) {
         if (Keys[j]==0) strcpy(sName,"*");
         else
           gdxSymbolInfo(Tptr, Keys[j], sName, &AUser2, &ATyp2);
         if (j < ADim-1) printf("%s,",sName);
         else printf("%s)",sName);
       }
     }
     if (strlen(sText)) printf(" '%s'", sText);
   }
   if (0 == ACount) {
     if (0 == ADim && GMS_DT_PAR == ATyp) /* Scalar */
       printf(" / 0.0 /;n");
     else if (GMS_DT_ALIAS != ATyp)
       printf(" / /;n");
   } else {
     printf(" /n");
     gdxDataReadRawStart (Tptr, i, &NRec);
     while (gdxDataReadRaw(Tptr,Keys,Vals,&FDim)) {
       if ((GMS_DT_VAR == ATyp || GMS_DT_EQU == ATyp) && 0 ==
memcmp(Vals,dv,GMS_VAL_MAX*sizeof(double))) /* all default records */
         continue;
       if (GMS_DT_PAR == ATyp && 0.0 == Vals[GMS_VAL_LEVEL])
         continue;
       for (j=1; j<=ADim; j++) {
         if (1==gdxUMUelGet(Tptr, Keys[j-1], UelName, &IDum))
           printf("'%s'", UelName);
         else {
           printf("L__",Keys[j-1]); BadUels++;
         }
         if (j < ADim) printf (".");
       }
       if (GMS_DT_PAR == ATyp)
         printf(" %sn", val2str(Tptr, Vals[GMS_VAL_LEVEL], msg));
       else if (GMS_DT_SET == ATyp)
         if (Vals[GMS_VAL_LEVEL]) {
           j = (int) Vals[GMS_VAL_LEVEL];
           gdxGetElemText(Tptr, j, msg, &IDum);
           printf(" '%s'n", msg);
         } else  printf("n");
       else if (GMS_DT_VAR == ATyp || GMS_DT_EQU == ATyp) {
         printf(" ."); c='(';
         for (j=GMS_VAL_LEVEL; j<GMS_VAL_MAX; j++) {
           if (Vals[j] != dv[j]) {
             if (GMS_VAL_SCALE == j && GMS_DT_VAR == ATyp &&
                 AUser != GMS_VARTYPE_POSITIVE && AUser != GMS_VARTYPE_NEGATIVE && AUser !=
GMS_VARTYPE_FREE)
               printf("%c prior %s", c, val2str(Tptr, Vals[GMS_VAL_SCALE], msg));
             else
               printf("%c %s %s", c, gmsValTypeText[j]+1, val2str(Tptr, Vals[j], msg));
             if ( '(' == c) c = ',';
           }
         }
         printf(" )n");
       }
     }
   }
   printf("/;n");
   j=1; while (gdxSymbolGetComment(Tptr, i, j++, msg)) printf("* %sn", msg);
   if (GMS_DT_VAR == ATyp || GMS_DT_EQU == ATyp) printf("$offtextn");
   printf("n");
 }
 printf("$offempty offembedded n");

 if (BadUels > 0)
   printf("**** %d reference(s) to unique elements without a string representationn", BadUels);

 gdxFree(&Tptr);

}
```

1.6.3 Example 3: C++ program

This is a simplified version of the gdxdump program written in C++

```
/*
  Use this command to compile the example:
  cl xp_example1.cpp api/gdxco.cpp ../C/api/gdxcc.c -Iapi -I../C/api
*/

#include <string>
#include <cstring>
#include <cstdlib>
#include <iostream>
#include "gdxco.hpp"

using namespace std;
using namespace GAMS;

static std::string Indx[GMS_MAX_INDEX_DIM];
static gdxValues_t Values;

void ReportGDXError(GDX &PGX) {
  std::string S;

  cout << "**** Fatal GDX Error" << endl;
  PGX.ErrorStr(PGX.GetLastError(), S);
  cout << "**** " << S << endl;
  exit(1);
}

void ReportIOError(int N, const std::string &msg) {
  cout << "**** Fatal I/O Error = " << N << " when calling " << msg << endl;
  exit(1);
}

void WriteData(GDX &PGX, const std::string &s, const double V) {
  Indx[0] = s;
  Values[GMS_VAL_LEVEL] = V;
  PGX.DataWriteStr(Indx,Values);
}

int main (int argc, char *argv[]) {

  std::string Msg, FileName, Producer, Sysdir, VarName;
  int         ErrNr;
  int         VarNr;
  int         NrRecs;
  int         N;
  int         Dim;
  int         VarTyp;
```

```
    int          D;

    if (argc < 2 || argc > 3) {
      cout << "**** xp_Example1: incorrect number of parameters" << endl;
      exit(1);
    }

    Sysdir = argv[1];
    cout << "xp_Example1 using GAMS system directory: " << Sysdir << endl;

    GDX PGX(Sysdir, Msg);

    if (Msg != "") {
      cout << "**** Could not load GDX library" << endl << "**** " << Msg << endl;
      exit(1);
    }

    PGX.GetDLLVersion(Msg);
    cout << "Using GDX DLL version: " << Msg << endl;

    if (2 == argc) {
      /* Write demand data */
      PGX.OpenWrite("demanddata.gdx", "xp_example1", ErrNr);
      if (ErrNr) ReportIOError(ErrNr,"gdxOpenWrite");
      if (!PGX.DataWriteStrStart("Demand","Demand data",1,GMS_DT_PAR ,0))
        ReportGDXError(PGX);
      WriteData(PGX,"New-York",324.0);
      WriteData(PGX,"Chicago" ,299.0);
      WriteData(PGX,"Topeka"  ,274.0);
      if (!PGX.DataWriteDone()) ReportGDXError(PGX);
      cout << "Demand data written by example1" << endl;
    } else {
      FileName = argv[2];
      PGX.OpenRead(FileName, ErrNr);
      if (ErrNr) ReportIOError(ErrNr,"gdxOpenRead");
      PGX.FileVersion(Msg,Producer);
      cout << "GDX file written using version: " << Msg << endl;
      cout << "GDX file written by: " <<  Producer << endl;

      if (!PGX.FindSymbol("x",VarNr)) {
        cout << "**** Could not find variable X" << endl;
        exit(1);
      }

      PGX.SymbolInfo(VarNr,VarName,Dim,VarTyp);
      if (Dim != 2 || GMS_DT_VAR != VarTyp) {
        cout << "**** X is not a two dimensional variable: "
             << Dim << ":" << VarTyp << endl;
        exit(1);
      }

      if (!PGX.DataReadStrStart(VarNr,NrRecs)) ReportGDXError(PGX);
      cout << "Variable X has " << NrRecs << " records" << endl;
      while (PGX.DataReadStr(Indx,Values,N)) {
        if (0 == Values[GMS_VAL_LEVEL]) continue; /* skip level 0.0 is default */
        for (D=0; D<Dim; D++) cout << (D? '.':' ') <<  Indx[D];
        cout << " = " << Values[GMS_VAL_LEVEL] << endl;
      }
      cout << "All solution values shown" << endl;
      PGX.DataReadDone();
    }

    if (ErrNr = PGX.Close()) ReportIOError(ErrNr,"gdxClose");

    return 0;
  } /* main */
```

1.6.4 Example 4: VB.NET program

This program has the same functionality as the Delphi program in Example1; see GAMS and Delphi (☐ see Example 1 in Delphi, page 12).

```
Module xp_example1
    '///////////////////////////////////////////////////////////////
    '// This program generates demand data for a modified version //
    '// of the trnsport model or reads the solution back from a   //
    '// gdx file.                                                 //
    '//                                                           //
    '// Calling convention:                                       //
    '// Case 1:                                                   //
    '//    Parameter 1: GAMS system directory                     //
    '// The program creates a GDX file with demand data           //
    '// Case 2:                                                   //
    '//    Parameter 1: GAMS system directory                     //
    '//    Parameter 2: gdxfile                                   //
    '// The program reads the solution from the GDX file          //
    '// Paul van der Eijk Jun-12, 2002                            //
    '///////////////////////////////////////////////////////////////

    Dim PGX As IntPtr

    Sub ReportGDXError(ByVal PGX As IntPtr)
        Dim S As String = String.Empty
        Console.WriteLine("**** Fatal GDX Error")
        gdxErrorStr(0, gdxGetLastError(PGX), S)
        Console.WriteLine("**** " & S)
        End
    End Sub

    Sub ReportIOError(ByVal N As Integer)
        Console.WriteLine("**** Fatal I/O Error = " & N)
        End
    End Sub

    Sub WriteData(ByVal s As String, ByVal V As Double)
        Dim Indx(maxdim) As String 'TgdxStrIndex
        Dim Values(val_max) As Double 'TgdxValues
        Indx(0) = s
        Values(val_level) = V
        gdxDataWriteStr(PGX, Indx, Values)
    End Sub


    Dim Msg As String
    Dim Sysdir As String
    Dim Producer As String
    Dim ErrNr, rc As Integer
    Dim Indx(maxdim) As String 'TgdxStrIndex
    Dim Values(val_max) As Double 'TgdxValues
    Dim VarNr As Integer
    Dim NrRecs As Integer
    Dim N As Integer
    Dim Dimen As Integer
    Dim VarName As String
    Dim VarTyp As Integer

    Sub Main()
        If Environment.GetCommandLineArgs().Length <> 2 And
Environment.GetCommandLineArgs().Length <> 3 Then
            Console.WriteLine("**** XP_Example1: incorrect number of parameters")
            End
        End If

        Sysdir = Environment.GetCommandLineArgs(1)
        Console.WriteLine("XP_Example1 using GAMS system directory: " & Sysdir)

        If Not gdxCreateD(PGX, Sysdir, Msg) Then
            Console.WriteLine("**** Could not load GDX library")
            Console.WriteLine("**** " & Msg)
            Exit Sub
```

```
          End If

          gdxGetDLLVersion(PGX, Msg)
          Console.WriteLine("Using GDX DLL version: " & Msg)

          If Environment.GetCommandLineArgs().Length = 2 Then
              'write demand data
              gdxOpenWrite(PGX, "demanddata.gdx", "xp_example1", ErrNr)
              If ErrNr <> 0 Then
                  ReportIOError(ErrNr)
              End If
              If gdxDataWriteStrStart(PGX, "Demand", "Demand data", 1, dt_par, 0) = 0 Then
                  ReportGDXError(PGX)
              End If
              WriteData("New-York", 324.0)
              WriteData("Chicago", 299.0)
              WriteData("Topeka", 274.0)
              If gdxDataWriteDone(PGX) = 0 Then
                  ReportGDXError(PGX)
              End If
              Console.WriteLine("Demand data written by xp_example1")
          Else
              rc = gdxOpenRead(PGX, Environment.GetCommandLineArgs(2), ErrNr)
  'Environment.GetCommandLineArgs(1) "trnsport.gdx"
              If ErrNr <> 0 Then
                  ReportIOError(ErrNr)
              End If

              'read x variable back (non-default level values only)
              gdxFileVersion(PGX, Msg, Producer)
              Console.WriteLine("GDX file written using version: " & Msg)
              Console.WriteLine("GDX file written by: " & Producer)

              If gdxFindSymbol(PGX, "x", VarNr) = 0 Then
                  Console.WriteLine("**** Could not find variable X")
                  Exit Sub
              End If

              gdxSymbolInfo(PGX, VarNr, VarName, Dimen, VarTyp)
              If (Dimen <> 2) Or (VarTyp <> dt_var) Then
                  Console.WriteLine("**** X is not a two dimensional variable")
                  Exit Sub
              End If

              If gdxDataReadStrStart(PGX, VarNr, NrRecs) = 0 Then
                  ReportGDXError(PGX)
              End If

              Console.WriteLine("Variable X has " & NrRecs & " records")
              While gdxDataReadStr(PGX, Indx, Values, N) <> 0
                  If Values(val_level) = 0.0 Then      'skip level = 0.0 is default
                      Continue While
                  End If
                  For D = 1 To Dimen
                      Console.Write(Indx(D - 1))
                      If D < Dimen Then
                          Console.Write(".")
                      End If
                  Next
                  Console.WriteLine(" = " & Values(val_level))
              End While
              Console.WriteLine("All solution values shown")
              gdxDataReadDone(PGX)
          End If

          ErrNr = gdxClose(PGX)
          If ErrNr <> 0 Then
              ReportIOError(ErrNr)
          End If
```

```
      End Sub

  End Module
```

1.6.5 Example 5: Fortran program

This program has the same functionality as the Delphi program in Example1; see GAMS and Delphi (see Example 1 in Delphi, page 12).

```
! To compile this example run:
! > ifort -c api/gdxf9def.f90
! > cl -DAPIWRAP_LCASE_NODECOR -c api/gdxf9glu.c -Iapi -I../C/api
! > lib -out:gdxf90lib.lib gdxf9def.obj gdxf9glu.obj
! > ifort -c api/gamsglobals_mod.f90 xp_example1.f90
! > ifort -exe:xp_example1.exe gamsglobals_mod.obj xp_example1.obj gdxf90lib.lib

MODULE exData
  USE gamsglobals
  IMPLICIT NONE
  CHARACTER(LEN=UEL_IDENT_LEN), DIMENSION(MAX_INDEX_DIM) :: Indx
  REAL(KIND=8), DIMENSION(val_max) :: Values
END MODULE exData

PROGRAM xp_example1
  USE gdxf9def
  USE gamsglobals
  USE exData
  IMPLICIT NONE

  LOGICAL             :: ok
  INTEGER(KIND=8)     :: PGX = 0
  INTEGER(KIND=4)     :: RC, ErrNr, VarNr, NrRecs, N, Dim, VarTyp, D, argc, iargc
  CHARACTER(LEN=255) :: Msg, Producer, Sysdir, VarName, gdxFname

  argc = iargc()

  IF ((argc /= 1) .AND. (argc /= 2)) THEN
    WRITE(*,*) '**** xp_Example1: incorrect number of parameters'
    CALL gdxExit(1)
  END IF

  CALL getarg(1, Sysdir)
  WRITE(*,*) 'xp_Example1 using GAMS system directory: ', Sysdir

  ok = gdxCreateD(PGX, Sysdir, Msg)

  IF (.NOT. ok) THEN
    WRITE(*,*) '**** Could not load GDX library'
    WRITE(*,*) '**** ', Msg
    CALL gdxExit(1)
  END IF

  RC = gdxGetDLLVersion(PGX, Msg)
  WRITE(*,*) 'Using GDX DLL version: ', Msg

  IF (1 == argc) THEN
!   Write demand data
    RC = gdxOpenWrite(PGX, './demanddata.gdx', 'example1', ErrNr)
    IF (ErrNr /= 0) CALL ReportIOError(ErrNr,'gdxOpenWrite')
    ok = 0 .ne. gdxDataWriteStrStart(PGX,'Demand','Demand data',1,DT_PAR ,0)
    IF (.NOT. ok) CALL ReportGDXError(PGX)
    CALL WriteData(PGX,'New-York',324D0)
    CALL WriteData(PGX,'Chicago',299D0)
    CALL WriteData(PGX,'Topeka',274D0)
    ok = 0 .ne. gdxDataWriteDone(PGX)
    IF (.NOT. ok) CALL ReportGDXError(PGX)
    WRITE(*,*) 'Demand data written by xp_example1'
  ELSE
!   Read variable X
    CALL getarg(2, gdxFname)
```

```
      RC = gdxOpenRead(PGX, gdxFname, ErrNr)
      IF (ErrNr /= 0) CALL ReportIOError(ErrNr,'gdxOpenRead')
      RC = gdxFileVersion(PGX,Msg,Producer)
      WRITE(*,*) 'GDX file written using version: ',Msg
      WRITE(*,*) 'GDX file written by: ',Producer

      ok = 0 .ne. gdxFindSymbol(PGX,'x',VarNr)
      IF (.NOT. ok) THEN
        WRITE(*,*) '**** Could not find variable X'
        CALL gdxExit(1)
      END IF

      RC = gdxSymbolInfo(PGX,VarNr,VarName,Dim,VarTyp)
      IF (Dim /= 2 .OR. DT_VAR /= VarTyp) THEN
        WRITE(*,*) '**** X is not a two dimensional variable: ',Dim,':',VarTyp
        CALL gdxExit(1)
      END IF

      ok = 0 .ne. gdxDataReadStrStart(PGX,VarNr,NrRecs)
      IF (.NOT. ok) CALL ReportGDXError(PGX)
      WRITE(*,*) 'Variable X has ',NrRecs,' records'
      DO WHILE (0 .ne. gdxDataReadStr(PGX,Indx,Values,N))
        IF (0D0 == Values(VAL_LEVEL)) CYCLE ! skip, level 0.0 is default
        DO D = 1,Dim
           IF (D /= DIM) THEN
              WRITE(*,*) Indx(D) ,'.'
           ELSE
              WRITE(*,*) Indx(D)
           END IF
        END DO
        write(*,*) ' = ', Values(VAL_LEVEL)
      END DO
      WRITE(*,*) 'All solution values shown'
      RC = gdxDataReadDone(PGX)
    END IF

    ErrNr = gdxClose(PGX)
    IF (ErrNr /= 0) CALL ReportIOError(ErrNr,'gdxClose')

    ok = gdxFree(PGX)
    IF (.NOT. ok) THEN
      WRITE(*,*) 'Problems unloading the GDX DLL'
      CALL gdxExit(1)
    END IF


  CONTAINS
    SUBROUTINE ReportGDXError(PGX)
      INTEGER(KIND=8), INTENT(IN) :: PGX
      CHARACTER(LEN=256) :: S
      WRITE (*,*) '**** Fatal GDX Error'
      RC = gdxErrorStr(PGX, gdxGetLastError(PGX), S)
      WRITE (*,*) '**** ', S
      STOP
    END SUBROUTINE ReportGDXError

    SUBROUTINE ReportIOError(N, msg)
      INTEGER(KIND=4), INTENT(IN) :: N
      CHARACTER(LEN=*), INTENT(IN) :: msg
      WRITE(*,*) '**** Fatal I/O Error = ', N, ' when calling ', msg
      STOP
    END SUBROUTINE ReportIOError

    SUBROUTINE WriteData(PGX, S, V)
      INTEGER(KIND=8), INTENT(IN) :: PGX
      CHARACTER(LEN=*), INTENT(IN) :: S
      REAL(KIND=8), INTENT(IN) :: V
      Indx(1) = S
      Values(VAL_LEVEL) = V
      RC = gdxDataWriteStr(PGX,Indx,Values)
    END SUBROUTINE WriteData

  END PROGRAM xp_example1
```

1.6.6 Example 6: Python program

This program has the same functionality as the Delphi program in Example1; see GAMS and Delphi (see Example 1 in Delphi, page 12).

```
from gdxcc import *
import sys
import os

numberParams = len(sys.argv)
if numberParams < 2 or numberParams > 3:
    print "**** Usage:", sys.argv[0], "sysDir [gdxinfn]"
    os._exit(1)

print sys.argv[0], "using GAMS system directory:", sys.argv[1]

gdxHandle = new_gdxHandle_tp()
rc =  gdxCreateD(gdxHandle, sys.argv[1], GMS_SSSIZE)
assert rc[0],rc[1]

print "Using GDX DLL version: " + gdxGetDLLVersion(gdxHandle)[1]

if numberParams == 2:
    assert gdxOpenWrite(gdxHandle, "demanddata.gdx", "xp_example1")[0]
    assert gdxDataWriteStrStart(gdxHandle, "Demand", "Demand data", 1, GMS_DT_PAR , 0)

    values = doubleArray(GMS_VAL_MAX)

    values[GMS_VAL_LEVEL] = 324.0
    gdxDataWriteStr(gdxHandle, ["New-York"], values)
    values[GMS_VAL_LEVEL] = 299.0
    gdxDataWriteStr(gdxHandle, ["Chicago"], values)
    values[GMS_VAL_LEVEL] = 274.0
    gdxDataWriteStr(gdxHandle, ["Topeka"], values)

    assert gdxDataWriteDone(gdxHandle)
    print "Demand data written by xp_example1"
else:
    assert gdxOpenRead(gdxHandle, sys.argv[2])[0]

    ret, fileVersion, producer = gdxFileVersion(gdxHandle)
    print "GDX file written using version: "+fileVersion
    print "GDX file written by: "+producer

    ret, symNr = gdxFindSymbol(gdxHandle, "x")
    assert ret, "Symbol x not found"

    ret, symName, dim, symType = gdxSymbolInfo(gdxHandle, symNr)
    assert dim == 2 and symType == GMS_DT_VAR, "**** x is not a two dimensional variable:n" +
"dim = " + str(dim) + "nvarTyp = " + str(symType)

    ret, nrRecs =  gdxDataReadStrStart(gdxHandle, symNr)
    assert ret, "Error in gdxDataReadStrStart:
"+gdxErrorStr(gdxHandle,gdxGetLastError(gdxHandle))[1]

    print "Variable x has", nrRecs, "records"
    for i in range(nrRecs):
        ret, elements, values, afdim = gdxDataReadStr(gdxHandle)
        assert ret, "Error in gdxDataReadStr:
"+gdxErrorStr(gdxHandle,gdxGetLastError(gdxHandle))[1]
        if 0 == values[GMS_VAL_LEVEL]: continue
        for d in range(dim):
            print elements[d],
            if d < dim-1:
                print ".",
        print " =", values[GMS_VAL_LEVEL]
    print "All solution values shown"
    gdxDataReadDone(gdxHandle)

assert not gdxClose(gdxHandle)
assert gdxFree(gdxHandle)

print "All done xp_example1"
```

1.6.7 Example 7: C# program

This program has the same functionality as the Delphi program in Example1; see GAMS and Delphi (see Example 1 in Delphi, page 12).

Note that the CSharp sub-directory of the apiexamples directory contains many more examples.

```
ï»¿///////////////////////////////////////////////////////////////
// This program generates demand data for a modified version //
// of the trnsport model or reads the solution back from a   //
// gdx file.                                                 //
//                                                           //
// Calling convention:                                       //
// Case 1:                                                   //
//    Parameter 1: GAMS system directory                     //
// The program creates a GDX file with demand data           //
// Case 2:                                                   //
//    Parameter 1: GAMS system directory                     //
//    Parameter 2: gdxfile                                   //
// The program reads the solution from the GDX file          //
// Paul van der Eijk Jun-12, 2002                            //
///////////////////////////////////////////////////////////////

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace xp_example1
{
    class xp_example1
    {
        static gdxcs gdx;
        static void ReportGDXError()
        {
            string S = string.Empty;
            Console.WriteLine("**** Fatal GDX Error");
            gdx.gdxErrorStr(gdx.gdxGetLastError(), ref S);
            Console.WriteLine("**** " + S);
            Environment.Exit(1);
        }

        static void ReportIOError(int N)
        {
            Console.WriteLine("**** Fatal I/O Error = " + N);
            Environment.Exit(1);
        }

        static void WriteData(string s, double V)
        {
            string[] Indx = new string[gamsglobals.maxdim];
            double[] Values = new double[gamsglobals.val_max];
            Indx[0] = s;
            Values[gamsglobals.val_level] = V;
            gdx.gdxDataWriteStr(Indx, Values);
        }

        static int Main(string[] args)
        {
            string Msg = string.Empty;
            string Sysdir;
            string Producer = string.Empty;
            int ErrNr = 0;
            int rc;
            string[] Indx = new string[gamsglobals.maxdim];
            double[] Values = new double[gamsglobals.val_max];
            int VarNr = 0;
            int NrRecs = 0;
            int N = 0;
            int Dimen = 0;
            string VarName = string.Empty;
            int VarTyp = 0;
            int D;

            if (Environment.GetCommandLineArgs().Length != 2 &&
  Environment.GetCommandLineArgs().Length != 3)
            {
                Console.WriteLine("**** XP_Example1: incorrect number of parameters");
                return 1;
            }

            String[] arguments = Environment.GetCommandLineArgs();
            Sysdir = arguments[1];
            Console.WriteLine("XP_Example1 using GAMS system directory: " + Sysdir);

            gdx = new gdxcs(Sysdir, ref Msg);
            if (Msg != string.Empty)
            {
                Console.WriteLine("**** Could not load GDX library");
                Console.WriteLine("**** " + Msg);
                return 1;
            }

            gdx.gdxGetDLLVersion(ref Msg);
            Console.WriteLine("Using GDX DLL version: " + Msg);

            if (Environment.GetCommandLineArgs().Length == 2)
            {
                //write demand data
                gdx.gdxOpenWrite("demanddata.gdx", "xp_example1", ref ErrNr);
                if (ErrNr != 0) xp_example1.ReportIOError(ErrNr);
                if (gdx.gdxDataWriteStrStart("Demand", "Demand data", 1, gamsglobals.dt_par,
  0) == 0) ReportGDXError();
                WriteData("New-York", 324.0);
                WriteData("Chicago", 299.0);
                WriteData("Topeka", 274.0);
                if (gdx.gdxDataWriteDone() == 0) ReportGDXError();
                Console.WriteLine("Demand data written by xp_example1");
            }
            else
            {
                rc = gdx.gdxOpenRead(arguments[2], ref ErrNr);
                if (ErrNr != 0) ReportIOError(ErrNr);

                //read x variable back (non-default level values only)
                gdx.gdxFileVersion(ref Msg, ref Producer);
                Console.WriteLine("GDX file written using version: " + Msg);
                Console.WriteLine("GDX file written by: " + Producer);

                if (gdx.gdxFindSymbol("x", ref VarNr) == 0)
                {
                    Console.WriteLine("**** Could not find variable X");
                    return 1;
                }

                gdx.gdxSymbolInfo(VarNr, ref VarName, ref Dimen, ref VarTyp);
                if (Dimen != 2 || VarTyp != gamsglobals.dt_var)
                {
                    Console.WriteLine("**** X is not a two dimensional variable");
                    return 1;
                }

                if (gdx.gdxDataReadStrStart(VarNr, ref NrRecs) == 0) ReportGDXError();

                Console.WriteLine("Variable X has " + NrRecs + " records");
                while (gdx.gdxDataReadStr(ref Indx, ref Values, ref N) != 0)
                {
                    if(Values[gamsglobals.val_level] == 0.0) //skip level = 0.0 is default
                        continue;
                    for (D=0; D<Dimen; D++)
                    {
                        Console.Write(Indx[D]);
                        if (D < Dimen-1) Console.Write(".");
                    }
                    Console.WriteLine(" = " + Values[gamsglobals.val_level]);
                }
                Console.WriteLine("All solution values shown");
                gdx.gdxDataReadDone();
            }
            ErrNr = gdx.gdxClose();
            if (ErrNr != 0) ReportIOError(ErrNr);
            return 0;
        }
    }
}
```

1.6.8 Example 8: Java program

This program has the same functionality as the Delphi program in Example1; see GAMS and Delphi (see Example 1 in Delphi, page 12).

Note that the Java sub-directory of the apiexamples directory contains many more examples.

```
package com.gams.xp_examples;
import com.gams.api.*;

public class xp_example1 {

static gdx gdxio = new gdx();
static String[] Indx = new String[gamsglobals.maxdim];
static double[] Values = new double[gamsglobals.val_max];

static void ReportGDXError() {
  String[] S = new String[1];

  System.out.println("**** Fatal GDX Error");
  gdxio.ErrorStr(gdxio.GetLastError(), S);
  System.out.println("**** " + S[0]);
  System.exit(1);
}

static void ReportIOError(int N, String msg ) {
  System.out.println("**** Fatal I/O Error = " + N + " when calling " + msg);
  System.exit(1);
}

static void WriteData(String s, double V) {
  Indx[0] = s;
  Values[gamsglobals.val_level] = V;
  gdxio.DataWriteStr(Indx,Values);
}

 public static void main (String[] args) {

  String[]    Msg = new String[1];
  String[]    Producer = new String[1];
  String      Sysdir;
  int[]       ErrNr = new int[1];
  int[]       VarNr = new int[1];
  int[]       NrRecs = new int[1];
```

```
    int[]       N = new int[1];
    int[]       Dim = new int[1];
    String[]    VarName = new String[1];
    int[]       VarTyp = new int[1];
    int         D;

    if (args.length < 1 || args.length > 2) {
      System.out.println("**** Example1: incorrect number of parameters");
      System.exit(1);
    }

    Sysdir = args[0];
    System.out.println("Example1 using GAMS system directory: " + Sysdir);

    if (gdxio.CreateD(Sysdir, Msg) != 1) {
      System.out.println("**** Could not load GDX library");
      System.out.println("**** " + Msg[0]);
      System.exit(1);
    }

    gdxio.GetDLLVersion(Msg);
    System.out.println("Using GDX DLL version: " + Msg[0]);

    if (1 == args.length) {
      /* Write demand data */
      gdxio.OpenWrite("demanddata.gdx", "example1", ErrNr);
      if (ErrNr[0] != 0) ReportIOError(ErrNr[0],"gdxOpenWrite");
      if (gdxio.DataWriteStrStart("Demand","Demand data",1,gamsglobals.dt_par,0) != 1)
        ReportGDXError();
      WriteData("New-York",324.0);
      WriteData("Chicago" ,299.0);
      WriteData("Topeka"  ,274.0);
      if (gdxio.DataWriteDone() != 1) ReportGDXError();
      System.out.println("Demand data written by example1n");
    } else {
      gdxio.OpenRead(args[1], ErrNr);
      if (ErrNr[0] != 0) ReportIOError(ErrNr[0],"gdxOpenRead");
      gdxio.FileVersion(Msg,Producer);
      System.out.println("GDX file written using version: " + Msg[0]);
      System.out.println("GDX file written by: " + Producer[0]);

      if (gdxio.FindSymbol("x",VarNr) != 1) {
        System.out.println("**** Could not find variable X");
        System.exit(1);
      }

      gdxio.SymbolInfo(VarNr[0],VarName,Dim,VarTyp);
      if (Dim[0] != 2 || gamsglobals.dt_var != VarTyp[0]) {
        System.out.println("**** X is not a two dimensional variable: " + Dim[0] + ":" +
  VarTyp[0]);
        System.exit(1);
      }

      if (gdxio.DataReadStrStart(VarNr[0],NrRecs) != 1) ReportGDXError();
      System.out.println("Variable X has " + NrRecs[0] + " records");
      while (gdxio.DataReadStr(Indx,Values,N) != 0) {
        if (0 == Values[gamsglobals.val_level]) continue; /* skip level 0.0 is default */
        for (D=0; D<Dim[0]; D++) {
            System.out.print(Indx[D]);
            if (D<Dim[0]-1) System.out.print(".");
        }
        System.out.println(" = " + Values[gamsglobals.val_level]);
      }
      System.out.println("All solution values shown");
      gdxio.DataReadDone();
    }

    ErrNr[0] = gdxio.Close();
    if (ErrNr[0] != 0) ReportIOError(ErrNr[0],"gdxClose");
    if (gdxio.Free() != 1) {
      System.out.println("Problems unloading the GDX DLL");
      System.exit(1);
    }

  } /* main */

  }
```

1.7 Conversion issues when moving from GAMS 22.5 to 22.6

- maximum number of dimensions = 20 (was 10)
- maximum length of an identifier or unique element = 63 (was 31)
- support for acronyms
- support for domain information

Backward compatibility:

- GAMS and all gdx utilites will write gdx files in the new format
- GAMS and all gdx utilites can read older gdx formats
- The gdxcopy utility can convert between different gdx formats (assuming that dimension and namelength is supported)

Libraries:

- gdxio.dll is still available but the new library is called gdxdclib.dll (substitute .dll with the extension for your platform)
- gdxio.dll cannot read the new gdx format

API:

- Functions in the library that used to return a boolean, now return an integer (zero for false, non-zero for true)
- Before we can read or write a gdx file, we need to create a valid gdx object. The function gdxCreate (see page 77) will create such an object
- The functions gdxOpenRead (see page 90) and gdxOpenWrite (see page 90) no longer create the gdx object pointer, they require an object pointer that has been initialized using gdxCreate (see page 77) or similar functions

1.8 Files in the apifiles directory

The following sections describe the various files included in the apifiles directory. All functions will use the gdxdclib library (like gdxdclib.dll on Windows). The entry points in the library can be loaded static (by the operating system) or dynamic. Dynamic loading provides more control when an entry point is missing or the interface has changed. Static loading will cause an exception to be generated for example for a missing entry point without much feedback about the error.

For Delphi/Pascal two different interfaces are available; an object interface and a function interface.

- C files (see page 29)
- Delphi/Pascal files (see page 30)
- Fortran files (see page 30)
- Java files (see page 31)
- VB files (see page 31)

1.8.1 C files

Subdir	File	Loading	Remarks
common	gamsglobals.h		Global constants
common	gamsglobals.cs		Global constants

common	gclgms.c		Global constants
common	gclgms.h		Global constants
examples	example1.c		Sample program C
examples	example1.cpp		Sample program C++
gdx	gdxcc.c	Dynamic	C
gdx	gdxcc.h	Dynamic	C
gdx	gdxcs.cs	Static	C#
gdx	gdxco.cpp	Dynamic	C++
gdx	gdxco.hpp	Dynamic	C++

1.8.2 Delphi/Pascal files

Subdir	File	Interface	Loading	Remarks
common	gmsgen.pas			Shared types
common	gmsspecs.pas			Special values
common	gxdefs.pas (see page 101)			Shared types
common	gxdefsp.pas			Shared types / Windows only
examples	example1.dpr	Function	Dynamic	Sample program
examples	example1do.dpr	Object	Stat/Dyn	Sample program
examples	example1dp.dpr	Function	Static	Sample program
gdx	gdxdcdef.pas	Function	Dynamic	
gdx	gdxdcon.pas			Shared constants
gdx	gdxdcpdef.pas	Function	Dynamic	Windows only
gdx	gdxddec.inc			
gdx	gdxdocpdef.pas	Object	Dynamic	Windows only
gdx	gdxdodef.pas	Object	Dynamic	
gdx	gdxdopdef.pas	Object	Static	
gdx	gdxdpdef.pas	Function	Static	Windows only

1.8.3 Fortran files

Subdir	File	Loading	Remarks
gdx	gdxf9def.f90	Dynamic	
gdx	gdxf9glu.c	Dynamic	

1.8.4 Java files

Subdir	File	Loading	Remarks
common	gamsglobals.java		Global constants
examples	example1.java	Static	Sample program Java
gdx	gdxjava.java	Static	
gdx	gdxjni.c	Dynamic	Java Native Interface

1.8.5 VB files

Subdir	File	Loading	Remarks
common	gamsglobals.bas		Global constants
common	gamsglobals.vb		Global constants
examples	example1.vb	Static	Sample program VB.Net
gdx	gdxvba.bas	Static	VBA
gdx	gdxvbnet.vb	Static	VB.Net

2 Symbol Reference

These are all symbols available in this documentation.

2.1 Classes

These are all classes that are contained in this documentation.

2.1.1 TGXFileObj

```
TGXFileObj = class
```

Class Hierarchy

TObject
  TGXFileObj (see page 32)

Unit

gxfile (see gxfile.pas, page 102)

TGXFileObj Members

Methods

Create	*Creates a gdx data object.*	Destroy	*Destroy the object*
gdxAcronymAdd	*Add a new acronym entry*	gdxAcronymCount	*Number of entries in the acronym table*
gdxAcronymGetInfo	*Retrieve acronym information from the acronym table*	gdxAcronymGetMapping	*Get information how acronym values are remapped*
gdxAcronymIndex	*Get index value of an acronym*	gdxAcronymName	*Find the name of an acronym value*
gdxAcronymNextNr	*Returns the value of the NextAutoAcronym variable and sets the variable to nv*	gdxAcronymSetInfo	*Modify acronym information in the acronym table*
gdxAcronymValue	*Create (see page 33) an acronym value based on the index*	gdxAddAlias	*Add an alias for a set to the symbol table*
gdxAddSetText	*Register a string in the string table*	gdxAutoConvert	*Returns the value of the AutoConvert variable and sets the variable to nv*
gdxClose	*Close a gdx file*	gdxCurrentDim	*Returns the dimension of the current active symbol*
gdxDataErrorCount	*The number of error records*	gdxDataErrorRecord	*Retrieve an error record*
gdxDataReadDone	*Finish reading of a symbol in any mode(raw, mapped, string)*	gdxDataReadFilteredStart	*Initialize the reading of a symbol in filtered mode*
gdxDataReadMap	*Read the next record in mapped mode*	gdxDataReadMapStart	*Initialize the reading of a symbol in mapped mode*
gdxDataReadRaw	*Read the next record in raw mode*	gdxDataReadRawFast	*Read a symbol in Raw mode using a callback procedure*
gdxDataReadRawFastFilt	*Read a symbol in Raw mode while applying a filter using a callback procedure*	gdxDataReadRawStart	*Initialize the reading of a symbol in raw mode*
gdxDataReadSlice	*Read a slice of data from a data set*	gdxDataReadSliceStart	*Prepare for the reading of a slice of data from a data set*
gdxDataReadStr	*Read the next record in string mode*	gdxDataReadStrStart	*Initialize the reading of a symbol in string mode*
gdxDataSliceUELS	*Map a slice index in to the corresponding unique elements*	gdxDataWriteDone	*Finish a write operation*
gdxDataWriteMap	*Write a data element in mapped mode*	gdxDataWriteMapStart	*Start writing a new symbol in mapped mode*
gdxDataWriteRaw	*Write a data element in raw mode*	gdxDataWriteRawStart	*Start writing a new symbol in raw mode*
gdxDataWriteStr	*Write a data element in string mode*	gdxDataWriteStrStart	*Start writing a new symbol in string mode*
gdxErrorCount	*Returns the number of errors*	gdxErrorStr	*Returns the text for a given error number*
gdxFileInfo	*Returns file format number and compression level used*	gdxFileVersion	*Return strings for file version and file producer*
gdxFilterExists	*Check if there is a filter defined based on its number*	gdxFilterRegister	*Add a unique element to the current filter definition*
gdxFilterRegisterDone	*Finish registration of unique elements for a filter*	gdxFilterRegisterStart	*Define a unique element filter*
gdxFindSymbol	*Find symbol by name*	gdxGetDLLVersion	*Returns a version descriptor of the library*
gdxGetDomainElements	*Get the unique elements for a given dimension of a given symbol*	gdxGetElemText	*Retrieve the string and node number for an entry in the string table*

gdxGetLastError	*Return the last error*
gdxGetMemoryUsed	*Return the number of bytes used by the data objects*
gdxGetSpecialValues	*Retrieve the internal values for special values*
gdxGetUEL	*Get the string for a unique element using a mapped index*
gdxMapValue	*Classify a value as a potential special value*
gdxOpenAppend	*Open an existing gdx file for output*
gdxOpenRead	*Open a gdx file for reading*
gdxOpenWrite	*Open a new gdx file for output; uses the environment variable GDXCOMPRESS to set compression argument for gdxOpenWriteEx (see page 55)*
gdxOpenWriteEx	*Create (see page 33) a gdx file for writing*
gdxRenameUEL	*Rename UEL OldName to NewName*
gdxResetSpecialValues	*Reset the internal values for special values*
gdxSetHasText	*Test if any of the elements of the set has an associated text*
gdxSetReadSpecialValues	*Set the internal values for special values when reading a gdx file*
gdxSetSpecialValues	*Set the internal values for special values*
gdxSetTextNodeNr	*Set the Node number for an entry in the string table*
gdxSetTraceLevel	*Set the amount of trace (debug) information generated*
gdxSymbIndxMaxLength	*Returns the length of the longest UEL used for every index position for a given symbol*
gdxSymbMaxLength	*Returns the length of the longest symbol name*
gdxSymbolAddComment	*Add a line of comment text for a symbol*
gdxSymbolDim	*Returns Dimension of a symbol*
gdxSymbolGetComment	*Retrieve a line of comment text for a symbol*
gdxSymbolGetDomain	*Retrieve the domain of a symbol*
gdxSymbolGetDomainX	*Retrieve the domain of a symbol (using relaxed or domain information)*
gdxSymbolInfo	*Returns information about a symbol*
gdxSymbolInfoX	*Returns additional information about a symbol*
gdxSymbolSetDomain	*Define the domain of a symbol*
gdxSymbolSetDomainX	*Define the domain of a symbol (relaxed version)*
gdxSystemInfo	*Returns the number of symbols and unique elements*
gdxUELMaxLength	*Returns the length of the longest UEL name*
gdxUELRegisterDone	*Finish registration of unique elements*
gdxUELRegisterMap	*Register an unique elements in mapped mode*
gdxUELRegisterMapStart	*Start registering unique elements in mapped mode*
gdxUELRegisterRaw	*Register an unique elements in raw mode*
gdxUELRegisterRawStart	*Start registering unique elements in raw mode*
gdxUELRegisterStr	*Register a unique element in string mode*
gdxUELRegisterStrStart	*Start registering unique elements in string mode*
gdxUMFindUEL	*Search for unique element by its string*
gdxUMUelGet	*Get a unique element using an unmapped index*
gdxUMUelInfo	*Return information about the unique elements*

Legend

▲virtual

Description

Class for reading and writing gdx files

TGXFileObj.Create

Creates a gdx data object.

```
constructor Create(var ErrMsg: ShortString);
```

Parameters

`var ErrMsg: ShortString`

Contains error message if any, or empty if there was no error

See Also

TGXFileObj.gdxOpenRead (see page 54), TGXFileObj.gdxOpenWrite (see page 54), TGXFileObj.gdxOpenWriteEx (see page 55)

TGXFileObj.Destroy

Destroy the object

```
destructor Destroy; override;
```

Return Value

None

Description

No pending write operations will be finished but the file will be closed. After closing the file, the object is freed.

TGXFileObj.gdxAcronymAdd

Add a new acronym entry

```
function gdxAcronymAdd(const AName: ShortString; const Txt: ShortString; AIndx: integer):
integer;
```

Parameters

`const AName: ShortString`

Name of the acronym

`const Txt: ShortString`

Explanatory text of the acronym

`AIndx: integer`

Index value of the acronym

Return Value

0 If the entry is not added because of a duplicate name using the same value fo the indx -1 If the entry is not added because of a duplicate name using a different value for the indx Otherwise the index into the acronym table (1..gdxAcronymCount (see TGXFileObj.gdxAcronymCount, page 34))

Description

This function can be used to add entries before data is written. When entries are added implicitly use gdxAcronymSetInfo (see TGXFileObj.gdxAcronymSetInfo, page 36) to update the table.

See Also

TGXFileObj.gdxAcronymGetInfo (see page 34), TGXFileObj.gdxAcronymCount (see page 34)

TGXFileObj.gdxAcronymCount

Number of entries in the acronym table

```
function gdxAcronymCount: integer;
```

Return Value

The number of entries in the acronym table

See Also

TGXFileObj.gdxAcronymSetInfo (see page 36), TGXFileObj.gdxAcronymSetInfo (see page 36)

TGXFileObj.gdxAcronymGetInfo

Retrieve acronym information from the acronym table

```
function gdxAcronymGetInfo(N: integer; var AName: ShortString; var Txt: ShortString; var
AIndx: integer): integer;
```

Parameters

`N: integer`

Index into acronym table; range from 1 to AcronymCount

`var AName: ShortString`

Name of the acronym

`var Txt: ShortString`

Explanatory text of the acronym

`var AIndx: integer`

Index value of the acronym

Return Value

Non-zero if the index into the acronym table is valid; false otherwise

See Also

TGXFileObj.gdxAcronymSetInfo (see page 36), TGXFileObj.gdxAcronymCount (see page 34)

TGXFileObj.gdxAcronymGetMapping

Get information how acronym values are remapped

```
function gdxAcronymGetMapping(N: integer; var orgIndx: integer; var newIndx: integer; var
autoIndex: integer): integer;
```

Parameters

`N: integer`

Index into acronym table; range from 1 to AcronymCount

`var orgIndx: integer`

The Index used in the gdx file

`var newIndx: integer`

The Index returned when reading gdx data

`var autoIndex: integer`

non-zero if the newIndx was generated using the value of NextAutoAcronym

Return Value

Non-zero if the index into the acronym table is valid; false otherwise

Description

When reading gdx data, we need to map indices for acronyms used in the gdx file to indices used by the reading program. There is a problen when not all acronyms have been registered before reading the gdx data. We need to map an udefined index we read to a new value. The value of NextAutoAcronym is used for that.

See Also

TGXFileObj.gdxAcronymGetInfo (see page 34), TGXFileObj.gdxAcronymCount (see page 34), TGXFileObj.gdxAcronymNextNr (see page 36)

TGXFileObj.gdxAcronymIndex

Get index value of an acronym

```
function gdxAcronymIndex(V: double): integer;
```

Parameters

`V: double`

Input value possibly representing an acronym

Return Value

Index of acronym value V; zero if V does not represent an acronym

See Also

TGXFileObj.gdxAcronymValue (see page 36)

TGXFileObj.gdxAcronymName

Find the name of an acronym value

```
function gdxAcronymName(V: double; var AName: ShortString): integer;
```

Parameters

`V: double`

Input value possibly containing an acronym `var AName: ShortString`

Name of acronym value or the empty string

Return Value

Return non-zero if a name for the acronym is defined. Return zero if V does not represent an acronym value or a name is not defined. An unnamed acronym value will return a string of the form UnknownAcronymNNN; were NNN is the index of the acronym.

See Also

TGXFileObj.gdxAcronymIndex (⧉ see page 35)

TGXFileObj.gdxAcronymNextNr

Returns the value of the NextAutoAcronym variable and sets the variable to nv

```
function gdxAcronymNextNr(nv: integer): integer;
```

Parameters

`nv: integer`

New value for NextAutoAcronym; a value of less than zero is ignored

Return Value

Previous value of NextAutoAcronym

Description

When we read from a gdx file and encounter an acronym that was not defined, we need to assign a new index for that acronym. The variable NextAutoAcronym is used for this purpose and is incremented for each new undefined acronym. When NextAutoAcronym has a value of zero, the default, the value is ignored and the original index as stored in the gdx file is used for the index.

TGXFileObj.gdxAcronymSetInfo

Modify acronym information in the acronym table

```
function gdxAcronymSetInfo(N: integer; const AName: ShortString; const Txt: ShortString;
AIndx: integer): integer;
```

Parameters

`N: integer`

Index into acronym table; range from 1 to AcronymCount

`const AName: ShortString`

Name of the acronym

`const Txt: ShortString`

Explanatory text of the acronym

`AIndx: integer`

Index value of the acronym

Return Value

Non-zero if the index into the acronym table is valid; false otherwise

Description

When writing a gdx file, this function is used to provide the name of an acronym; in this case the Indx parameter must match. When reading a gdx file, this function is used to provide the acronym index, and the AName parameter must match.

See Also

TGXFileObj.gdxAcronymGetInfo (⧉ see page 34), TGXFileObj.gdxAcronymCount (⧉ see page 34)

TGXFileObj.gdxAcronymValue

Create (⧉ see TGXFileObj.Create, page 33) an acronym value based on the index

```
function gdxAcronymValue(AIndx: integer): double;
```

Parameters

`AIndx: integer`

Index value; should be greater than zero

Return Value

The calculated acronym value; zero if Indx is not positive

See Also

TGXFileObj.gdxAcronymIndex (see page 35)

TGXFileObj.gdxAddAlias

Add an alias for a set to the symbol table

```
function gdxAddAlias(const Id1: ShortString; const Id2: ShortString): integer;
```

Parameters

`AName1`

set identifier

`AName2`

set identifier

Return Value

Non-zero if the operation is possible, zero otherwise

Description

One of the two identifiers has to be a known set, an alias or * (universe); the other identifier is used as the new alias for the given set. The function gdxSymbolInfoX (see TGXFileObj.gdxSymbolInfoX, page 60) can be used to retrieve the set or alias associated with the identifier; it is returned as the UserInfo parameter.

See Also

TGXFileObj.gdxSymbolSetDomain (see page 60)

TGXFileObj.gdxAddSetText

Register a string in the string table

```
function gdxAddSetText(const Txt: ShortString; var TxtNr: integer): integer;
```

Parameters

`const Txt: ShortString`

The string to be registered

`var TxtNr: integer`

The index number assigned to this string

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Register a string in the string table and return the integer number assigned to this string. The integer value can be used to set the associated text of a set element. The string must follow the GAMS syntax rules for explanatory text.

See Also

TGXFileObj.gdxGetElemText (see page 51), TGXFileObj.gdxSetTextNodeNr (see page 57)

TGXFileObj.gdxAutoConvert

Returns the value of the AutoConvert variable and sets the variable to nv

```
function gdxAutoConvert(nv: integer): integer;
```

Parameters

```
nv: integer
```

New value for AutoConvert

Return Value

Previous value of AutoConvert

Description

When we close a new gdx file, we look at the value of AutoConvert; if AutoConvert is non-zero, we look at the GDXCOMPRESS and GDXCONVERT environment variables to determine if conversion to an older file format is desired. We needed this logic so gdxcopy.exe can disable automatic file conversion.

TGXFileObj.gdxClose

Close a gdx file

```
function gdxClose: integer;
```

Return Value

Returns the value of gdxGetLastError (see TGXFileObj.gdxGetLastError, page 52)

Description

Close a gdx file that was previously opened for reading or writing. Before the file is closed, any pending write operations will be finished. To free the gdx object, call gdxFree (see page 86).

See Also

TGXFileObj.gdxOpenRead (see page 54), TGXFileObj.gdxOpenWrite (see page 54)

TGXFileObj.gdxCurrentDim

Returns the dimension of the current active symbol

```
function gdxCurrentDim: Integer;
```

Return Value

Dimension of current active symbol

Description

When reading or writing data, the dimension of the current active symbol is sometimes needed to convert arguments from strings to pchars etc.

TGXFileObj.gdxDataErrorCount

The number of error records

```
function gdxDataErrorCount: integer;
```

Return Value

The number of error records available.

Description

After a write operation is finished (gdxDataWriteDone (see TGXFileObj.gdxDataWriteDone, page 45)), the data is sorted and written to the gdx file. If there are duplicate records, the first record is written to the file and the duplicates are added to the error list.

When reading data using a filtered read operation, data records that were filtered out because an index is not in the user index space or not in a filter are added the error list.

See Also

TGXFileObj.gdxDataErrorRecord (see page 38)

TGXFileObj.gdxDataErrorRecord

Retrieve an error record

```
function gdxDataErrorRecord(RecNr: integer; var KeyInt: TgdxUELIndex; var Values: TgdxValues):
```

```
integer;
```

Parameters

`RecNr: integer`

The number of the record to be retrieved, range = 1..NrErrorRecords

`var KeyInt: TgdxUELIndex`

Index for the record

`var Values: TgdxValues`

Values for the record

Return Value

Non-zero if the record number is valid, zero otherwise

See Also

TGXFileObj.gdxDataErrorCount (see page 38)

TGXFileObj.gdxDataReadDone

Finish reading of a symbol in any mode(raw, mapped, string)

```
function gdxDataReadDone: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataReadRawStart (see page 42), TGXFileObj.gdxDataReadMapStart (see page 40), TGXFileObj.gdxDataReadStrStart (see page 44)

TGXFileObj.gdxDataReadFilteredStart

Initialize the reading of a symbol in filtered mode

```
function gdxDataReadFilteredStart(SyNr: integer; const FilterAction: TgdxUELIndex; var NrRecs:
integer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 0..NrSymbols; SyNr = 0 reads universe

`const FilterAction: TgdxUELIndex`

Array of filter actions for each index position

`var NrRecs: integer`

The maximum number of records available for reading. The actual number of records may be less when a filter is applied to the records read.

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Start reading data for a symbol in filtered mode. Each filter action (1..Dimension) describes how each index should be treated when reading a data record. When new unique elements are returned, they are added to the user index space automatically. The actual reading of records is done with DataReadMap.

The action codes are as follows:

Action code	Result
DOMC_UNMAPPED (see page 100)	The index is not mapped into user space

DOMC_EXPAND (see page 99)	New unique elements encountered will be be mapped into the user space
DOMC_STRICT (see page 100)	If the unique element in this position does not map into user space, the record will not be available and is added to the error list instead
FilterNumber	If the unique element in this position does not map into user space or is not enabled in this filter, the record will not be available and is added to the error list instead

See Also

TGXFileObj.gdxFilterRegisterStart (see page 49), TGXFileObj.gdxDataReadMap (see page 40), TGXFileObj.gdxDataReadRawStart (see page 42), TGXFileObj.gdxDataReadStrStart (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadMap

Read the next record in mapped mode

```
function gdxDataReadMap(RecNr: integer; var KeyInt: TgdxUELIndex; var Values: TgdxValues; var
DimFrst: integer): integer;
```

Parameters

`RecNr: integer`

Ignored (left in for backward compatibility)

`var KeyInt: TgdxUELIndex`

The index of the record

`var Values: TgdxValues`

The data of the record

`var DimFrst: integer`

The first index position in KeyInt that changed

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataReadMapStart (see page 40), TGXFileObj.gdxDataReadFilteredStart (see page 39), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadMapStart

Initialize the reading of a symbol in mapped mode

```
function gdxDataReadMapStart(SyNr: integer; var NrRecs: integer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 0..NrSymbols; SyNr = 0 reads universe

`var NrRecs: integer`

The number of records available for reading

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataReadMap (see page 40), TGXFileObj.gdxDataReadRawStart (see page 42), TGXFileObj.gdxDataReadStrStart (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadRaw

Read the next record in raw mode

```
function gdxDataReadRaw(var KeyInt: TgdxUELIndex; var Values: TgdxValues; var DimFrst:
integer): integer;
```

Parameters

`var KeyInt: TgdxUELIndex`

The index of the record

`var Values: TgdxValues`

The data of the record

`var DimFrst: integer`

The first index position in KeyInt that changed

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataReadRawStart (see page 42), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadRawFast

Read a symbol in Raw mode using a callback procedure

```
function gdxDataReadRawFast(SyNr: integer; DP: TDataStoreProc; var NrRecs: integer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 0..NrSymbols; SyNr = 0 reads universe

`DP: TDataStoreProc`

Procedure that will be called for each data record

`var NrRecs: integer`

The number of records available for reading

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Use a callback function to read a symbol in raw mode. Using a callback procedure to read the data is faster because we no longer have to check the context for each call to read a record.

See Also

TGXFileObj.gdxDataReadRaw (see page 41), TGXFileObj.gdxDataReadMapStart (see page 40), TGXFileObj.gdxDataReadStrStart (see page 44), TGXFileObj.gdxDataReadDone (see page 39), TGXFileObj.gdxDataReadRawFastFilt (see page 41)

TGXFileObj.gdxDataReadRawFastFilt

Read a symbol in Raw mode while applying a filter using a callback procedure

```
function gdxDataReadRawFastFilt(SyNr: integer; const UelFilterStr: TgdxStrIndex; DP:
TDataStoreFiltProc): integer;
```

Parameters

`const UelFilterStr: TgdxStrIndex`

Each index can be fixed by setting the string for the unique element. Set an index position to the empty string in order not to fix that position.

`DP: TDataStoreFiltProc`

Callback procedure which will be called for each available data item

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Read a slice of data, by fixing zero or more index positions in the data. When a data element is available, the callback procedure DP is called with the current index (as raw numbers) and the values.

See Also

TGXFileObj.gdxDataReadRawFast (see page 41), TGXFileObj.gdxDataReadSliceStart (see page 43), TGXFileObj.gdxDataSliceUELS (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

Examples

Example

```
function DPCallBack(const Indx: TgdxUELIndex; const Vals: TgdxValues; Uptr: Pointer): integer;
stdcall;
var
   s: ShortString;
   UelMap: integer;
begin
Result := 1;
gdxUMUelGet(Uptr, Indx[2], s, UelMap);
WriteLn(s, ' ', Vals[vallevel]);
end;

var
   pgx  : PGXFile;
   Msg  : ShortString;
   ErrNr: integer;
   IndxS: TgdxStrIndex;

IndxS[1] := 'i200'; IndxS[2] := '';
gdxDataReadRawFastFilt(pgx, 1, IndxS, DPCallBack);
```

TGXFileObj.gdxDataReadRawStart

Initialize the reading of a symbol in raw mode

```
function gdxDataReadRawStart(SyNr: integer; var NrRecs: integer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 0..NrSymbols; SyNr = 0 reads universe

`var NrRecs: integer`

The number of records available for reading

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataReadRaw (see page 41), TGXFileObj.gdxDataReadMapStart (see page 40), TGXFileObj.gdxDataReadStrStart (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadSlice

Read a slice of data from a data set

```
function gdxDataReadSlice(const UelFilterStr: TgdxStrIndex; var Dimen: integer; DP:
TDataStoreProc): integer;
```

Parameters

`const UelFilterStr: TgdxStrIndex`

Each index can be fixed by setting the string for the unique element. Set an index position to the empty string in order not to fix that position.

`var Dimen: integer`

The dimension of the index space; this is the number of index positions that is not fixed.

`DP: TDataStoreProc`

Callback procedure which will be called for each available data item

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Read a slice of data, by fixing zero or more index positions in the data. When a data element is available, the callback procedure DP is called with the current index and the values. The indices used in the index vary from zero to the highest value minus one for that index position. This function can be called multiple times.

See Also

TGXFileObj.gdxDataReadSliceStart (see page 43), TGXFileObj.gdxDataSliceUELS (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadSliceStart

Prepare for the reading of a slice of data from a data set

```
function gdxDataReadSliceStart(SyNr: integer; var ElemCounts: TgdxUELIndex): integer;
```

Parameters

`SyNr: integer`

Symbol number to read, range 1..NrSymbols

`var ElemCounts: TgdxUELIndex`

Array of integers, each position indicating the number of unique indices in that position

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Prepare for the reading of a slice of data. The actual read of the data is done by calling gdxDataReadSlice (see TGXFileObj.gdxDataReadSlice, page 42). When finished reading, call gdxDataReadDone (see TGXFileObj.gdxDataReadDone, page 39).

See Also

TGXFileObj.gdxDataReadSlice (see page 42), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadStr

Read the next record in string mode

```
function gdxDataReadStr(var KeyStr: TgdxStrIndex; var Values: TgdxValues; var DimFrst:
integer): integer;
```

Parameters

`var KeyStr: TgdxStrIndex`

The index of the record as strings for the unique elements

`var Values: TgdxValues`

The data of the record

`var DimFrst: integer`

The first index position in KeyStr that changed

Return Value

Non-zero if the operation is possible; return zero if the operation is not possible or if there is no more data

Description

Read the next record using strings for the unique elements. The reading should be initialized by calling DataReadStrStart

See Also

TGXFileObj.gdxDataReadStrStart (see page 44), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataReadStrStart

Initialize the reading of a symbol in string mode

```
function gdxDataReadStrStart(SyNr: integer; var NrRecs: integer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 0..NrSymbols; SyNr = 0 reads universe

`var NrRecs: integer`

The number of records available for reading

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Reading data using strings is the simplest way to read data. Every record read using DataReadStr will return the strings for the unique elements. Internal mapping is not affected by this function.

See Also

TGXFileObj.gdxDataReadStr (see page 43), TGXFileObj.gdxDataReadRawStart (see page 42), TGXFileObj.gdxDataReadMapStart (see page 40), TGXFileObj.gdxDataReadDone (see page 39)

Examples

Example

if DataReadStrStart(PGX,1,NrRecs)

then

begin

while DataReadStr(PGX,Uels,Vals)

do [...]

DataReadDone(PGX)

end;

TGXFileObj.gdxDataSliceUELS

Map a slice index in to the corresponding unique elements

```
function gdxDataSliceUELS(const SliceKeyInt: TgdxUELIndex; var KeyStr: TgdxStrIndex): integer;
```

Parameters

`const SliceKeyInt: TgdxUELIndex`

The slice index to be mapped to strings.

`var KeyStr: TgdxStrIndex`

Array of strings containg the unique elements

Return Value

Non-zero if the operation is possible, zero otherwise

Description

After calling DataReadSliceStart, each index position is mapped from 0 to N(d)-1. This function maps this index space back in to unique elements represented as strings.

See Also

TGXFileObj.gdxDataReadSliceStart (see page 43), TGXFileObj.gdxDataReadDone (see page 39)

TGXFileObj.gdxDataWriteDone

Finish a write operation

```
function gdxDataWriteDone: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataErrorCount (see page 38), TGXFileObj.gdxDataWriteRawStart (see page 46), TGXFileObj.gdxDataWriteMapStart (see page 45), TGXFileObj.gdxDataWriteStrStart (see page 47)

TGXFileObj.gdxDataWriteMap

Write a data element in mapped mode

```
function gdxDataWriteMap(const KeyInt: TgdxUELIndex; const Values: TgdxValues): integer;
```

Parameters

`const KeyInt: TgdxUELIndex`

The index for this element using mapped values

`const Values: TgdxValues`

The values for this element

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataWriteMapStart (see page 45), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxDataWriteMapStart

Start writing a new symbol in mapped mode

```
function gdxDataWriteMapStart(const SyId: ShortString; const ExplTxt: ShortString; Dimen:
integer; Typ: integer; UserInfo: integer): integer;
```

Parameters

`const SyId: ShortString`

Name of the symbol

`const ExplTxt: ShortString`

Explanatory text for the symbol

`Dimen: integer`

Dimension of the symbol

`UserInfo: integer`

See gdxDataWriteRawStart (see TGXFileObj.gdxDataWriteRawStart, page 46) for more information

`Type`

Type of the symbol

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataWriteMap (see page 45), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxDataWriteRaw

Write a data element in raw mode

```
function gdxDataWriteRaw(const KeyInt: TgdxUELIndex; const Values: TgdxValues): integer;
```

Parameters

`const KeyInt: TgdxUELIndex`

The index for this element

`const Values: TgdxValues`

The values for this element

Return Value

Non-zero if the operation is possible, zero otherwise

Description

When writing data in raw mode, the index space used is based on the internal index space. The indices used are in the range 1..NrUels but this is not enforced. Before we can write in raw mode, the unique elements (strings) should be registered first.

When writing raw, it assumed that the records are written in sorted order and that there are no duplicate records. Records that are not in sorted order or are duplicates will be added to the error list (see DataErrorCount and DataErrorRecord)

See Also

TGXFileObj.gdxDataWriteRawStart (see page 46), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxDataWriteRawStart

Start writing a new symbol in raw mode

```
function gdxDataWriteRawStart(const SyId: ShortString; const ExplTxt: ShortString; Dimen:
integer; Typ: integer; UserInfo: integer): integer;
```

Parameters

`const SyId: ShortString`

Name of the symbol

`const ExplTxt: ShortString`

Explanatory text for the symbol

`Dimen: integer`

Dimension of the symbol

`Typ: integer`

Type of the symbol

`UserInfo: integer`

GAMS follows the following conventions:

Type	Value(s)
Aliased Set	The symbol number of the aliased set, or zero for the universe
Set	Zero
Parameter	Zero
Variable	The variable type: binary=1, integer=2, positive=3, negative=4, free=5, sos1=6, sos2=7, semicontinous=8, semiinteger=9
Equation	The equation type: eque=53, equg=54, equl=55, equn=56, equx=57, equc=58, equb=59

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataWriteRaw (see page 46), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxDataWriteStr

Write a data element in string mode

```
function gdxDataWriteStr(const KeyStr: TgdxStrIndex; const Values: TgdxValues): integer;
```

Parameters

`const KeyStr: TgdxStrIndex`

The index for this element using strings for the unique elements

`const Values: TgdxValues`

The values for this element

Return Value

Non-zero if the operation is possible, zero otherwise

Description

When writing data using string elements, each string element is added to the internal unique element table and assigned an index. Writing using strings does not add the unique elements to the user mapped space. Each element string must follow the GAMS rules for unique elements.

See Also

TGXFileObj.gdxDataWriteMapStart (see page 45), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxDataWriteStrStart

Start writing a new symbol in string mode

```
function gdxDataWriteStrStart(const SyId: ShortString; const ExplTxt: ShortString; Dimen:
integer; Typ: integer; UserInfo: integer): integer;
```

Parameters

`const SyId: ShortString`

Name of the symbol

`const ExplTxt: ShortString`

Explanatory text for the symbol

`Dimen: integer`

Dimension of the symbol

`Typ: integer`

Type of the symbol

`UserInfo: integer`

See gdxDataWriteRawStart (see TGXFileObj.gdxDataWriteRawStart, page 46) for more information

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxDataWriteStr (see page 47), TGXFileObj.gdxDataWriteDone (see page 45)

TGXFileObj.gdxErrorCount

Returns the number of errors

```
function gdxErrorCount: integer;
```

Return Value

Total number of errors encountered

See Also

TGXFileObj.gdxGetLastError (see page 52)

TGXFileObj.gdxErrorStr

Returns the text for a given error number

```
function gdxErrorStr(ErrNr: integer; var ErrMsg: ShortString): integer;
```

Parameters

`N`

Error number

`S`

Contains error text after return

Return Value

Always returns non-zero

See Also

TGXFileObj.gdxGetLastError (see page 52)

TGXFileObj.gdxFileInfo

Returns file format number and compression level used

```
function gdxFileInfo(var FileVer: integer; var ComprLev: integer): integer;
```

Parameters

`var FileVer: integer`

File format number or zero if the file is not open

`var ComprLev: integer`

Compression used; 0= no compression, 1=zlib

Return Value

Always returns non-zero

TGXFileObj.gdxFileVersion

Return strings for file version and file producer

```
function gdxFileVersion(var FileStr: ShortString; var ProduceStr: ShortString): integer;
```

Parameters

`var FileStr: ShortString`

Version string

`var ProduceStr: ShortString`

Producer string

Return Value

Always non-zero

Description

function gdxObsoleteFunction(const FuncName: ShortString): integer;

See Also

TGXFileObj.gdxOpenWrite (see page 54), TGXFileObj.gdxOpenWriteEx (see page 55)

TGXFileObj.gdxFilterExists

Check if there is a filter defined based on its number

```
function gdxFilterExists(FilterNr: integer): integer;
```

Parameters

`FilterNr: integer`

Filter number as used in FilterRegisterStart

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxFilterRegisterStart (see page 49)

TGXFileObj.gdxFilterRegister

Add a unique element to the current filter definition

```
function gdxFilterRegister(UelMap: integer): integer;
```

Parameters

`UelMap: integer`

Unique element number in the user index space

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Register a unique element as part of the current filter. The function returns false if the index number is out of range of valid user indices or the index was never mapped into the user index space.

See Also

TGXFileObj.gdxFilterRegisterStart (see page 49), TGXFileObj.gdxFilterRegisterDone (see page 49)

TGXFileObj.gdxFilterRegisterDone

Finish registration of unique elements for a filter

```
function gdxFilterRegisterDone: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxFilterRegisterStart (see page 49), TGXFileObj.gdxFilterRegister (see page 49)

TGXFileObj.gdxFilterRegisterStart

Define a unique element filter

```
function gdxFilterRegisterStart(FilterNr: integer): integer;
```

Parameters

`FilterNr: integer`

Filter number to be assigned

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Start the registration of a filter. A filter is used to map a number of elements to a single integer; the filter number. A filter number can later be used to specify a filter for an index postion when reading data.

See Also

TGXFileObj.gdxFilterRegister (see page 49), TGXFileObj.gdxFilterRegisterDone (see page 49), TGXFileObj.gdxDataReadFilteredStart (see page 39)

TGXFileObj.gdxFindSymbol

Find symbol by name

```
function gdxFindSymbol(const SyId: ShortString; var SyNr: integer): integer;
```

Parameters

`const SyId: ShortString`

Name of the symbol

`var SyNr: integer`

Symbol number

Return Value

Non-zero if the symbol is found, zero otherwise.

Description

Search for a symbol by name; the search is not case sensitive. When the symbol is found, SyNr contains the symbol number and the function returns true. When the symbol is not found, the function returns false.

See Also

TGXFileObj.gdxSymbolInfo (see page 60), TGXFileObj.gdxSymbolInfoX (see page 60)

TGXFileObj.gdxGetDLLVersion

Returns a version descriptor of the library

```
function gdxGetDLLVersion(var V: ShortString): integer;
```

Parameters

`var V: ShortString`

Contains version string after return

Return Value

Always returns non-zero

TGXFileObj.gdxGetDomainElements

Get the unique elements for a given dimension of a given symbol

```
function gdxGetDomainElements(SyNr: integer; DimPos: integer; FilterNr: integer; DP:
TDomainIndexProc; var NrElem: integer; UPtr: pointer): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 1..NrSymbols

`DimPos: integer`

The dimension to use, range 1..dim

`FilterNr: integer`

Number of a previously registered filter or the value DOMC_EXPAND (see page 99) if no filter is wanted

`DP: TDomainIndexProc`

Callback procedure which will be called once for each available element (can be nil)

`var NrElem: integer`

Number of unique elements found

`UPtr: pointer`

User pointer; will be passed to the callback procedure

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Using the data of a symbol, get the unique elements for a given index position. To achieve this, the symbol's data is read and a tally is kept for the elements in the given index position. When a filter is specified, records that have elements in the specified index position that are outside the filter will be added to the list of DataErrorRecords. See gdxDataErrorRecord (see TGXFileObj.gdxDataErrorRecord, page 38)

See Also

gdxDataErrorCount gdxDataErrorRecord

Examples

Example

```
var
   T0 : Cardinal;
   Cnt: integer;

procedure DataDomainCB(RawNr, MappedNr: integer; UPtr: pointer); stdcall;
begin
Write(RawNr, ' (', MappedNr, ')');
end;

T0 := GetTickCount();
gdxGetDomainElements(PGX, 1, 1, DOMC_EXPAND, nil, cnt);
WriteLn('Domain count only = ',cnt ,' ', GetTickCount - T0, ' ms');
T0 := GetTickCount();
gdxGetDomainElements(PGX, 1, 1, DOMC_EXPAND, DataDomainCB, cnt);
WriteLn('Get domain count = ',cnt ,' ', GetTickCount - T0, ' ms');
T0 := GetTickCount();
gdxGetDomainElements(PGX, 1, 1, 7, DataDomainCB, cnt);
WriteLn('Using filter 7; number of records in error list = ', gdxDataErrorCount(PGX) );
```

TGXFileObj.gdxGetElemText

Retrieve the string and node number for an entry in the string table

```
function gdxGetElemText(TxtNr: integer; var Txt: ShortString; var Node: integer): integer;
```

Parameters

`TxtNr: integer`

String table index

`var Txt: ShortString`

Text found for the entry

`var Node: integer`

Node number found for the entry

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Retrieve a string based on the string table index. When writing to a gdx file, this index is the value returned by calling gdxAddSetText (see TGXFileObj.gdxAddSetText, page 37). When reading a gdx file, the index is returned as the level value when reading a set. The Node number can be used as an index in a string table in the user space; the value is set by calling SetTextNodeNr. If the Node number was never assigned, it will be returned as zero.

See Also

TGXFileObj.gdxAddSetText (see page 37), TGXFileObj.gdxSetTextNodeNr (see page 57)

Examples

Example

```
[assumes we are reading using strings ...]
while gdxDataReadStr(PGX, Uels, Vals) <> 0
do begin
   for D := 1 to Dim
   do Write(Uels[D], '  ');
   indx := Round(Vals[vallevel]);
   if indx > 0
   then
      begin
      gdxGetElemText(indx, S, N);
      Write('txt = ', S, ' Node = ', N);
      end;
   WriteLn;
   end
```

TGXFileObj.gdxGetLastError

Return the last error

```
function gdxGetLastError: integer;
```

Return Value

The error number, or zero if there was no error

Description

When an error is encountered, an error code is stored which can be retrieved with this function. If subsequent errors occur before this function is called, the first error code will be maintained. Calling this function will clear the last error stored.

See Also

TGXFileObj.gdxErrorCount (see page 47)

TGXFileObj.gdxGetMemoryUsed

Return the number of bytes used by the data objects

```
function gdxGetMemoryUsed: int64;
```

Return Value

The number of bytes used by the data objects

TGXFileObj.gdxGetSpecialValues

Retrieve the internal values for special values

```
function gdxGetSpecialValues(var Avals: TgdxSVals): integer;
```

Parameters

`var Avals: TgdxSVals`

array of special values used for Eps, +Inf, -Inf, NA and Undef

Return Value

Always non-zero

See Also

TGXFileObj.gdxResetSpecialValues (see page 56), TGXFileObj.gdxSetSpecialValues (see page 57)

TGXFileObj.gdxGetUEL

Get the string for a unique element using a mapped index

```
function gdxGetUEL(UelNr: integer; var Uel: ShortString): integer;
```

Parameters

`UelNr: integer`

Index number in user space (1..NrUserElem) `var Uel: ShortString`

String for the unique element

Return Value

Return non-zero if the index is in a valid range, zero otherwise

Description

Retrieve the string for an unique element based on a mapped index number.

See Also

TGXFileObj.gdxUMUelGet (see page 64)

TGXFileObj.gdxMapValue

Classify a value as a potential special value

```
function gdxMapValue(D: double; var sv: integer): integer;
```

Parameters

`D: double`

Value to classify

`var sv: integer`

Classification

Return Value

Returns non-zero if D is a special value, zero otherwise

See Also

TGXFileObj.gdxGetSpecialValues (see page 52), TGXFileObj.gdxSetSpecialValues (see page 57)

TGXFileObj.gdxOpenAppend

Open an existing gdx file for output

```
function gdxOpenAppend(const FileName: ShortString; const Producer: ShortString; var ErrNr:
integer): integer;
```

Parameters

`const FileName: ShortString`

File name of the gdx file to be created

`const Producer: ShortString`

Name of program that appends to the gdx file

`var ErrNr: integer`

Returns an error code or zero if there is no error

Return Value

Returns non-zero if the file can be opened; zero otherwise

Description

Open an existing gdx file for output. If a file extension is not supplied, the extension '.gdx' will be used. The return code is a system dependent I/O error. When appending to a gdx file, the symbol table, uel table etc will be read and the whole setup will be treated as if all sysmbols were just written to the gdx file. Replacing a symbol is not allowed; it will generate a duplicate symbol error.

See Also

TGXFileObj.gdxOpenRead (see page 54), TGXFileObj.gdxOpenWrite (see page 54), TGXFileObj.gdxOpenWriteEx (see page 55)

Examples

Example

```
var
   ErrNr: integer;
   PGX  : PGXFile;
   Msg  : ShortString;
begin
if not gdxGetReady(Msg)
then
   begin
   WriteLn('Cannot load GDX library, msg: ', Msg);
   exit;
   end;
gdxOpenAppend(PGX,'c:\mydata\file1.gdx','Examples', ErrCode);
if ErrCode <> 0
then
   [ ... ]
```

TGXFileObj.gdxOpenRead

Open a gdx file for reading

function gdxOpenRead(**const** FileName: ShortString; **var** ErrNr: integer): integer;

Parameters

`const FileName: ShortString`

file name of the gdx file to be opened

`var ErrNr: integer`

Returns an error code or zero if there is no error

Return Value

Returns non-zero if the file can be opened; zero otherwise

Description

Open an existing gdx file for input. If a file extension is not supplied, the extension '.gdx' will be used. The return code is a system dependent I/O error. If the file was found, but is not a valid gdx file, the function GetLastError can be used to handle these type of errors.

See Also

TGXFileObj.gdxOpenWrite (see page 54), TGXFileObj.Destroy (see page 33), TGXFileObj.gdxGetLastError (see page 52)

Examples

Example

```
var
   ErrNr: integer;
   PGX  : PGXFile;
begin
gdxOpenRead(PGX,'c:\mydata\file1.gdx', ErrNr);
if ErrNr <> 0
then
   begin
   [...]
```

TGXFileObj.gdxOpenWrite

Open a new gdx file for output; uses the environment variable GDXCOMPRESS to set compression argument for gdxOpenWriteEx (see TGXFileObj.gdxOpenWriteEx, page 55)

function gdxOpenWrite(**const** FileName: ShortString; **const** Producer: ShortString; **var** ErrNr: integer): integer;

Parameters

`const FileName: ShortString`

File name of the gdx file to be created

`const Producer: ShortString`

Name of program that creates the gdx file

`var ErrNr: integer`

Returns an error code or zero if there is no error

Return Value

Returns non-zero if the file can be opened; zero otherwise

Description

See gdxOpenWriteEx (see TGXFileObj.gdxOpenWriteEx, page 55)

See Also

TGXFileObj.gdxOpenRead (see page 54), TGXFileObj.gdxOpenWriteEx (see page 55), TGXFileObj.Destroy (see page 33)

TGXFileObj.gdxOpenWriteEx

Create (see TGXFileObj.Create, page 33) a gdx file for writing

```
function gdxOpenWriteEx(const FileName: ShortString; const Producer: ShortString; Compr:
integer; var ErrNr: integer): integer;
```

Parameters

`const FileName: ShortString`

File name of the gdx file to be created

`const Producer: ShortString`

Name of program that creates the gdx file

`Compr: integer`

Zero for no compression; non-zero uses compression if available Important! when writing compressed, set the AutoConvert flag to zero so the file is not uncompressed after the Close; see gdxAutoConvert (see TGXFileObj.gdxAutoConvert, page 37)

`var ErrNr: integer`

Returns an error code or zero if there is no error

Return Value

Returns non-zero if the file can be opened; zero otherwise

Description

Open a new gdx file for output. If a file extension is not supplied, the extension '.gdx' will be used. The return code is a system dependent I/O error.

See Also

TGXFileObj.gdxOpenRead (see page 54), TGXFileObj.gdxOpenWrite (see page 54), TGXFileObj.gdxAutoConvert (see page 37), TGXFileObj.Destroy (see page 33)

Examples

Example

```
var
   ErrNr: integer;
   PGX  : PGXFile;
   Msg  : ShortString;
begin
if not gdxGetReady(Msg)
then
   begin
   WriteLn('Cannot load GDX library, msg: ', Msg);
   exit;
   end;
```

```
    gdxOpenWriteEx(PGX,'c:\mydata\file1.gdx','Examples', 1, ErrCode);
    gdxAutoConvert(PGX, 0);
    if ErrCode <> 0
    then
       [ ... ]
```

TGXFileObj.gdxRenameUEL

Rename UEL OldName to NewName

```
function gdxRenameUEL(const OldName: ShortString; const NewName: ShortString): integer;
```

Parameters

`const OldName: ShortString`

Name of an existing UEL

`const NewName: ShortString`

New name for the UEL

Return Value

Zero if the renaming was possible; non-zero is an error indicator

TGXFileObj.gdxResetSpecialValues

Reset the internal values for special values

```
function gdxResetSpecialValues: integer;
```

Return Value

Always non-zero

See Also

TGXFileObj.gdxSetSpecialValues (see page 57), TGXFileObj.gdxGetSpecialValues (see page 52)

TGXFileObj.gdxSetHasText

Test if any of the elements of the set has an associated text

```
function gdxSetHasText(SyNr: integer): integer;
```

Parameters

`SyNr: integer`

Set Symbol number (1..NrSymbols)

Return Value

Non-zero if the Set contains at least one element that has associated text, zero otherwise

See Also

TGXFileObj.gdxSystemInfo (see page 61), TGXFileObj.gdxSymbolInfo (see page 60)

TGXFileObj.gdxSetReadSpecialValues

Set the internal values for special values when reading a gdx file

```
function gdxSetReadSpecialValues(const AVals: TgdxSVals): integer;
```

Parameters

`const AVals: TgdxSVals`

array of special values to be used for Eps, +Inf, -Inf, NA and Undef Note that the values do not have to be unique

Return Value

Always non-zero

Notes

Before calling this function, initialize the array of special values by calling gdxGetSpecialValues (see TGXFileObj.gdxGetSpecialValues, page 52) first

See Also

TGXFileObj.gdxSetSpecialValues (see page 57), TGXFileObj.gdxResetSpecialValues (see page 56), TGXFileObj.gdxGetSpecialValues (see page 52)

TGXFileObj.gdxSetSpecialValues

Set the internal values for special values

```
function gdxSetSpecialValues(const AVals: TgdxSVals): integer;
```

Parameters

`const AVals: TgdxSVals`

array of special values to be used for Eps, +Inf, -Inf, NA and Undef Note that the values have to be unique

Return Value

Non-zero if all values specified are unique, zero otherwise

Notes

Before calling this function, initialize the array of special values by calling gdxGetSpecialValues (see TGXFileObj.gdxGetSpecialValues, page 52) first

See Also

TGXFileObj.gdxSetReadSpecialValues (see page 56), TGXFileObj.gdxResetSpecialValues (see page 56), TGXFileObj.gdxGetSpecialValues (see page 52)

TGXFileObj.gdxSetTextNodeNr

Set the Node number for an entry in the string table

```
function gdxSetTextNodeNr(TxtNr: integer; Node: integer): integer;
```

Parameters

`TxtNr: integer`

Index number of the entry to be modified

`Node: integer`

The new Node value for the entry

Return Value

Non-zero if the operation is possible, zero otherwise

Description

After registering a string with AddSetText, we can assign a node number for later retrieval. The node number is any integer which is stored without further restrictions.

See Also

TGXFileObj.gdxAddSetText (see page 37), TGXFileObj.gdxGetElemText (see page 51)

TGXFileObj.gdxSetTraceLevel

Set the amount of trace (debug) information generated

```
function gdxSetTraceLevel(N: integer; const s: ShortString): integer;
```

Parameters

`N: integer`

Tracing level N <= 0 no tracing N >= 3 maximum tracing

`const s: ShortString`

A string to be included in the trace output

Return Value

Always non-zero

TGXFileObj.gdxSymbIndxMaxLength

Returns the length of the longest UEL used for every index position for a given symbol

```
function gdxSymbIndxMaxLength(SyNr: integer; var LengthInfo: TgdxUELIndex): integer;
```

Parameters

`SyNr: integer`

Symbol number

`var LengthInfo: TgdxUELIndex`

The longest length for each index position

Return Value

The length of the longest UEL found in the data

See Also

TGXFileObj.gdxUELMaxLength (see page 61)

TGXFileObj.gdxSymbMaxLength

Returns the length of the longest symbol name

```
function gdxSymbMaxLength: integer;
```

Return Value

The length of the longest symbol name

TGXFileObj.gdxSymbolAddComment

Add a line of comment text for a symbol

```
function gdxSymbolAddComment(SyNr: integer; const Txt: ShortString): integer;
```

Parameters

`SyNr: integer`

The symbol number (range 1..NrSymbols); if SyNr <= 0 the current symbol being written

`const Txt: ShortString`

String to add

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxSymbolGetComment (see page 59)

TGXFileObj.gdxSymbolDim

Returns Dimension of a symbol

```
function gdxSymbolDim(SyNr: integer): integer;
```

Parameters

`SyNr: integer`

The symbol number (range 0..NrSymbols); return universe info when SyNr = 0

Return Value

-1 if the symbol number is not in the correct range, the symbol's dimension otherwise

See Also

TGXFileObj.gdxSymbolInfo (see page 60), TGXFileObj.gdxSymbolInfoX (see page 60), TGXFileObj.gdxFindSymbol (see page 50)

TGXFileObj.gdxSymbolGetComment

Retrieve a line of comment text for a symbol

```
function gdxSymbolGetComment(SyNr: integer; N: integer; var Txt: ShortString): integer;
```

Parameters

`SyNr: integer`

The symbol number (range 1..NrSymbols)

`N: integer`

Line number (1..Count)

`var Txt: ShortString`

String containing the line requested

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxSymbolAddComment (see page 58)

TGXFileObj.gdxSymbolGetDomain

Retrieve the domain of a symbol

```
function gdxSymbolGetDomain(SyNr: integer; var DomainSyNrs: TgdxUELIndex): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 1..NrSymbols

`var DomainSyNrs: TgdxUELIndex`

array returning the set identifiers or *; DomainSyNrs[D] will contain the index number of the one dimensional set or alias used as the domain for index position D. A value of zero represents the universe (*)

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxSymbolSetDomain (see page 60), TGXFileObj.gdxSymbolGetDomainX (see page 59)

TGXFileObj.gdxSymbolGetDomainX

Retrieve the domain of a symbol (using relaxed or domain information)

```
function gdxSymbolGetDomainX(SyNr: integer; var DomainIDs: TgdxStrIndex): integer;
```

Parameters

`SyNr: integer`

The index number of the symbol, range 1..NrSymbols DomainIDs[D] will contain the strings as they were stored with the call gdxSymbolSetDomainX (see TGXFileObj.gdxSymbolSetDomainX, page 61). If gdxSymbolSetDomainX (see TGXFileObj.gdxSymbolSetDomainX, page 61) was never called, but gdxSymbolSetDomain (see TGXFileObj.gdxSymbolSetDomain, page 60) was called, that information will be used instead.

Return Value

0: If operation was not possible (Bad SyNr) 1: No domain information was available 2: Data used was defined using gdxSymbolSetDomainX (see TGXFileObj.gdxSymbolSetDomainX, page 61) 3: Data used was defined using gdxSymbolSetDomain (see TGXFileObj.gdxSymbolSetDomain, page 60)

See Also

TGXFileObj.gdxSymbolSetDomainX (see page 61), TGXFileObj.gdxSymbolSetDomain (see page 60)

TGXFileObj.gdxSymbolInfo

Returns information about a symbol

```
function gdxSymbolInfo(SyNr: integer; var SyId: ShortString; var Dimen: integer; var Typ:
integer): integer;
```

Parameters

`SyNr: integer`

The symbol number (range 0..NrSymbols); return universe info when SyNr = 0

`var SyId: ShortString`

Name of the symbol

`var Dimen: integer`

Dimension of the symbol

`var Typ: integer`

Symbol type

Return Value

Zero if the symbol number is not in the correct range, non-zero otherwise

See Also

TGXFileObj.gdxSystemInfo (see page 61), TGXFileObj.gdxSymbolInfoX (see page 60), TGXFileObj.gdxSymbolDim (see page 58), TGXFileObj.gdxFindSymbol (see page 50)

TGXFileObj.gdxSymbolInfoX

Returns additional information about a symbol

```
function gdxSymbolInfoX(SyNr: integer; var RecCnt: integer; var UserInfo: integer; var
ExplTxt: ShortString): integer;
```

Parameters

`SyNr: integer`

The symbol number (range 0..NrSymbols); return universe info when SyNr = 0

`var RecCnt: integer`

Total number of records stored (unmapped); for the universe (SyNr = 0) this is the number of entries when the gdx file was openened for reading.

`var UserInfo: integer`

User field value; see gdxDataWriteRawStart (see TGXFileObj.gdxDataWriteRawStart, page 46) for more information

`var ExplTxt: ShortString`

Explanatory text for the symbol

Return Value

Zero if the symbol number is not in the correct range, non-zero otherwise

See Also

TGXFileObj.gdxSystemInfo (see page 61), TGXFileObj.gdxSymbolInfo (see page 60), TGXFileObj.gdxFindSymbol (see page 50)

TGXFileObj.gdxSymbolSetDomain

Define the domain of a symbol

```
function gdxSymbolSetDomain(const DomainIDs: TgdxStrIndex): integer;
```

Parameters

`const DomainIDs: TgdxStrIndex`

array of identifers or *

Return Value

Non-zero if the operation is possible, zero otherwise

Description

This function defines the domain for the symbol for which a write data operation just started using DataWriteRawStart, DataWriteMapStart or DataWriteStrStart. At this point the symbol and dimension is known, but no data has been written yet. Each identifier will be checked to be a one dimensional set or an alias. When a domain is specified, write operations will be domain checked; records violating the domain will be added the the internal error list (see DataErrorCount and DataErrorRecord.)

See Also

TGXFileObj.gdxSymbolGetDomain (see page 59)

TGXFileObj.gdxSymbolSetDomainX

Define the domain of a symbol (relaxed version)

```
function gdxSymbolSetDomainX(SyNr: integer; const DomainIDs: TgdxStrIndex): integer;
```

Parameters

`const DomainIDs: TgdxStrIndex`

array of identifers or *

Return Value

Non-zero if the operation is possible, zero otherwise

Description

This function defines the relaxed domain information for the symbol SyNr. The identifiers will NOT be checked to be known one-dimensional sets, and no domain checking will be performed. This function can be called during or after the write operation. If domain checking is needed, use gdxSymbolSetDomain (see TGXFileObj.gdxSymbolSetDomain, page 60)

See Also

TGXFileObj.gdxSymbolSetDomain (see page 60), TGXFileObj.gdxSymbolGetDomainX (see page 59)

TGXFileObj.gdxSystemInfo

Returns the number of symbols and unique elements

```
function gdxSystemInfo(var SyCnt: integer; var UelCnt: integer): integer;
```

Parameters

`var SyCnt: integer`

Number of symbols available in the gdx file

`var UelCnt: integer`

Number of unique elements stored in the gdx file

Return Value

Returns a non-zero value

TGXFileObj.gdxUELMaxLength

Returns the length of the longest UEL name

```
function gdxUELMaxLength: integer;
```

Return Value

The length of the longest UEL name

See Also

TGXFileObj.gdxSymbIndxMaxLength (see page 58)

TGXFileObj.gdxUELRegisterDone

Finish registration of unique elements

```
function gdxUELRegisterDone: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxUELRegisterRawStart (see page 63), TGXFileObj.gdxUELRegisterMapStart (see page 62), TGXFileObj.gdxUELRegisterStrStart (see page 63)

TGXFileObj.gdxUELRegisterMap

Register an unique elements in mapped mode

```
function gdxUELRegisterMap(UMap: integer; const Uel: ShortString): integer;
```

Parameters

`UMap: integer`

User index number to be assigned to the unique element

`const Uel: ShortString`

String for unique element

Return Value

Non-zero if the operation is possible, zero otherwise

Description

Register a unique element in mapped space; UMap is the user assigned index for the element. Registering an element a second time is not considered an error as long as the same UMap is used. Assigning different elements with the same UMap value is an error. A unique element must follow the GAMS rules when it contains quote characters.

See Also

TGXFileObj.gdxUELRegisterMapStart (see page 62), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUELRegisterMapStart

Start registering unique elements in mapped mode

```
function gdxUELRegisterMapStart: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxUELRegisterMap (see page 62), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUELRegisterRaw

Register an unique elements in raw mode

```
function gdxUELRegisterRaw(const Uel: ShortString): integer;
```

Parameters

`const Uel: ShortString`

String for unique element

Return Value

Non-zero if the operation is possible, zero otherwise

Description

The unique element is registered in raw mode, i.e. the internally assigned integer index is determined by the system Can only be used while writing to a gdx file

See Also

TGXFileObj.gdxUELRegisterMap (see page 62), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUELRegisterRawStart

Start registering unique elements in raw mode

```
function gdxUELRegisterRawStart: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxUELRegisterRaw (see page 62), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUELRegisterStr

Register a unique element in string mode

```
function gdxUELRegisterStr(const Uel: ShortString; var UelNr: integer): integer;
```

Parameters

`const Uel: ShortString`

String for unique element

`var UelNr: integer`

Index number assigned to this unique element in user space

Return Value

Non-zero if the element was registered, zero otherwise.

Description

The unique element is registered in user mapped space. The returned index is the next higher value. Registering an element a second time is not considered an error and the same index position will be returned. A unique element must follow the GAMS rules when it contains quote characters.

See Also

TGXFileObj.gdxUELRegisterStrStart (see page 63), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUELRegisterStrStart

Start registering unique elements in string mode

```
function gdxUELRegisterStrStart: integer;
```

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxUELRegisterStr (see page 63), TGXFileObj.gdxUELRegisterDone (see page 61)

TGXFileObj.gdxUMFindUEL

Search for unique element by its string

```
function gdxUMFindUEL(const Uel: ShortString; var UelNr: integer; var UelMap: integer):
integer;
```

Parameters

`const Uel: ShortString`

String to be searched

`var UelNr: integer`

Internal unique element number or -1 if not found

`var UelMap: integer`

User mapping for the element or -1 if not found or the element was never mapped

Return Value

Non-zero if the element was found, zero otherwise

TGXFileObj.gdxUMUelGet

Get a unique element using an unmapped index

```
function gdxUMUelGet(UelNr: integer; var Uel: ShortString; var UelMap: integer): integer;
```

Parameters

`UelNr: integer`

Element number (unmapped) in the range 1..NrElem

`var Uel: ShortString`

String for unique element

`var UelMap: integer`

User mapping for this element or -1 if element was never mapped

Return Value

Non-zero if the operation is possible, zero otherwise

See Also

TGXFileObj.gdxUMUelInfo (see page 64), TGXFileObj.gdxGetUEL (see page 52)

TGXFileObj.gdxUMUelInfo

Return information about the unique elements

```
function gdxUMUelInfo(var UelCnt: integer; var HighMap: integer): integer;
```

Parameters

`var UelCnt: integer`

Total number of unique elements (uels in gdx file + new registered uels)

`var HighMap: integer`

Highest user mapping index used

Return Value

Always returns non-zero

See Also

TGXFileObj.gdxUMUelGet (see page 64)

2.2 Functions

These are all functions that are contained in this documentation.

2.2.1 BgdxDataReadStr

```
function BgdxDataReadStr(pgdx: pointer; var KeyStr: TgdxStrIndex; var Values: TgdxValues; var
DimFrst: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the VB wrapped version of TGXFileObj.gdxDataReadStr (see page 43)

2.2.2 BgdxDataSliceUELS

```
function BgdxDataSliceUELS(pgdx: pointer; const SliceKeyInt: TgdxUELIndex; var KeyStr:
TgdxStrIndex): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the VB wrapped version of TGXFileObj.gdxDataSliceUELS (see page 44)

2.2.3 BgdxSymbolGetDomainX

```
function BgdxSymbolGetDomainX(pgdx: pointer; SyNr: Integer; var DomainIDs: TgdxStrIndex):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the VB wrapped version of TGXFileObj.gdxSymbolGetDomainX (see page 59)

2.2.4 CgdxAcronymAdd

```
function CgdxAcronymAdd(pgdx: pointer; const AName: PAnsiChar; const Txt: PAnsiChar; AIndx:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAcronymAdd (see page 34)

2.2.5 CgdxAcronymGetInfo

```
function CgdxAcronymGetInfo(pgdx: pointer; N: Integer; AName: PAnsiChar; Txt: PAnsiChar; var
AIndx: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAcronymGetInfo (see page 34)

2.2.6 CgdxAcronymName

```
function CgdxAcronymName(pgdx: pointer; V: Double; AName: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (⧉ see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAcronymName (⧉ see page 35)

2.2.7 CgdxAcronymSetInfo

```
function CgdxAcronymSetInfo(pgdx: pointer; N: Integer; const AName: PAnsiChar; const Txt:
PAnsiChar; AIndx: Integer): Integer; stdcall;
```

Unit

gdxdclib (⧉ see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAcronymSetInfo (⧉ see page 36)

2.2.8 CgdxAddAlias

```
function CgdxAddAlias(pgdx: pointer; const Id1: PAnsiChar; const Id2: PAnsiChar): Integer;
stdcall;
```

Unit

gdxdclib (⧉ see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAddAlias (⧉ see page 37)

2.2.9 CgdxAddSetText

```
function CgdxAddSetText(pgdx: pointer; const Txt: PAnsiChar; var TxtNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (⧉ see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxAddSetText (⧉ see page 37)

2.2.10 CgdxDataReadRawFastFilt

```
function CgdxDataReadRawFastFilt(pgdx: pointer; SyNr: Integer; UelFilterStr: PPointerArray;
DP: TDataStoreFiltProc): Integer; stdcall;
```

Unit

gdxdclib (⧉ see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataReadRawFastFilt (see page 41)

2.2.11 CgdxDataReadSlice

```
function CgdxDataReadSlice(pgdx: pointer; UelFilterStr: PPointerArray; var Dimen: Integer; DP:
TDataStoreProc): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataReadSlice (see page 42)

2.2.12 CgdxDataReadStr

```
function CgdxDataReadStr(pgdx: pointer; KeyStr: PPointerArray; var Values: TgdxValues; var
DimFrst: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataReadStr (see page 43)

2.2.13 CgdxDataSliceUELS

```
function CgdxDataSliceUELS(pgdx: pointer; const SliceKeyInt: TgdxUELIndex; KeyStr:
PPointerArray): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataSliceUELS (see page 44)

2.2.14 CgdxDataWriteMapStart

```
function CgdxDataWriteMapStart(pgdx: pointer; const SyId: PAnsiChar; const ExplTxt: PAnsiChar;
Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataWriteMapStart (see page 45)

2.2.15 CgdxDataWriteRawStart

```
function CgdxDataWriteRawStart(pgdx: pointer; const SyId: PAnsiChar; const ExplTxt: PAnsiChar;
Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataWriteRawStart (see page 46)

2.2.16 CgdxDataWriteStr

```
function CgdxDataWriteStr(pgdx: pointer; KeyStr: PPointerArray; const Values: TgdxValues):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataWriteStr (see page 47)

2.2.17 CgdxDataWriteStrStart

```
function CgdxDataWriteStrStart(pgdx: pointer; const SyId: PAnsiChar; const ExplTxt: PAnsiChar;
Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxDataWriteStrStart (see page 47)

2.2.18 CgdxErrorStr

```
function CgdxErrorStr(pgdx: pointer; ErrNr: Integer; ErrMsg: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxErrorStr (see page 48)

2.2.19 CgdxFileVersion

```
function CgdxFileVersion(pgdx: pointer; FileStr: PAnsiChar; ProduceStr: PAnsiChar): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxFileVersion (see page 48)

2.2.20 CgdxFindSymbol

```
function CgdxFindSymbol(pgdx: pointer; const SyId: PAnsiChar; var SyNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxFindSymbol (see page 50)

2.2.21 CgdxGetDLLVersion

```
function CgdxGetDLLVersion(pgdx: pointer; V: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxGetDLLVersion (see page 50)

2.2.22 CgdxGetElemText

```
function CgdxGetElemText(pgdx: pointer; TxtNr: Integer; Txt: PAnsiChar; var Node: Integer):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxGetElemText (see page 51)

2.2.23 CgdxGetUEL

```
function CgdxGetUEL(pgdx: pointer; UelNr: Integer; Uel: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxGetUEL (see page 52)

2.2.24 CgdxOpenAppend

```
function CgdxOpenAppend(pgdx: pointer; const FileName: PAnsiChar; const Producer: PAnsiChar;
var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxOpenAppend (see page 53)

2.2.25 CgdxOpenRead

```
function CgdxOpenRead(pgdx: pointer; const FileName: PAnsiChar; var ErrNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxOpenRead (see page 54)

2.2.26 CgdxOpenWrite

```
function CgdxOpenWrite(pgdx: pointer; const FileName: PAnsiChar; const Producer: PAnsiChar;
var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxOpenWrite (see page 54)

2.2.27 CgdxOpenWriteEx

```
function CgdxOpenWriteEx(pgdx: pointer; const FileName: PAnsiChar; const Producer: PAnsiChar;
Compr: Integer; var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxOpenWriteEx (see page 55)

2.2.28 CgdxRenameUEL

```
function CgdxRenameUEL(pgdx: pointer; const OldName: PAnsiChar; const NewName: PAnsiChar):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxRenameUEL (see page 56)

2.2.29 CgdxSetTraceLevel

```
function CgdxSetTraceLevel(pgdx: pointer; N: Integer; const s: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSetTraceLevel (see page 57)

2.2.30 CgdxSymbolAddComment

```
function CgdxSymbolAddComment(pgdx: pointer; SyNr: Integer; const Txt: PAnsiChar): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolAddComment (see page 58)

2.2.31 CgdxSymbolGetComment

```
function CgdxSymbolGetComment(pgdx: pointer; SyNr: Integer; N: Integer; Txt: PAnsiChar):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolGetComment (see page 59)

2.2.32 CgdxSymbolGetDomainX

```
function CgdxSymbolGetDomainX(pgdx: pointer; SyNr: Integer; DomainIDs: PPointerArray):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolGetDomainX (see page 59)

2.2.33 CgdxSymbolInfo

```
function CgdxSymbolInfo(pgdx: pointer; SyNr: Integer; SyId: PAnsiChar; var Dimen: Integer; var
Typ: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolInfo (see page 60)

2.2.34 CgdxSymbolInfoX

```
function CgdxSymbolInfoX(pgdx: pointer; SyNr: Integer; var RecCnt: Integer; var UserInfo:
Integer; ExplTxt: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolInfoX (see page 60)

2.2.35 CgdxSymbolSetDomain

```
function CgdxSymbolSetDomain(pgdx: pointer; DomainIDs: PPointerArray): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolSetDomain (see page 60)

2.2.36 CgdxSymbolSetDomainX

```
function CgdxSymbolSetDomainX(pgdx: pointer; SyNr: Integer; DomainIDs: PPointerArray):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxSymbolSetDomainX (see page 61)

2.2.37 CgdxUELRegisterMap

```
function CgdxUELRegisterMap(pgdx: pointer; UMap: Integer; const Uel: PAnsiChar): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxUELRegisterMap (see page 62)

2.2.38 CgdxUELRegisterRaw

```
function CgdxUELRegisterRaw(pgdx: pointer; const Uel: PAnsiChar): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxUELRegisterRaw (see page 62)

2.2.39 CgdxUELRegisterStr

```
function CgdxUELRegisterStr(pgdx: pointer; const Uel: PAnsiChar; var UelNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxUELRegisterStr (see page 63)

2.2.40 CgdxUMFindUEL

```
function CgdxUMFindUEL(pgdx: pointer; const Uel: PAnsiChar; var UelNr: Integer; var UelMap:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxUMFindUEL (see page 63)

2.2.41 CgdxUMUelGet

```
function CgdxUMUelGet(pgdx: pointer; UelNr: Integer; Uel: PAnsiChar; var UelMap: Integer):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the C wrapped version of TGXFileObj.gdxUMUelGet (see page 64)

2.2.42 FgdxDataReadRawFastFilt

```
function FgdxDataReadRawFastFilt(pgdx: pointer; SyNr: Integer; const UelFilterStr:
TgdxStrIndex; DP: TDataStoreFiltProc_F): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadRawFastFilt (see page 41)

2.2.43 FgdxGetDomainElements

```
function FgdxGetDomainElements(pgdx: pointer; SyNr: Integer; DimPos: Integer; FilterNr:
Integer; DP: TDomainIndexProc_F; var NrElem: Integer; Uptr: Pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetDomainElements (see page 50)

2.2.44 gdxAcronymAdd

```
function gdxAcronymAdd(pgdx: pointer; const AName: ShortString; const Txt: ShortString; AIndx:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymAdd (see page 34)

2.2.45 gdxAcronymCount

```
function gdxAcronymCount(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymCount (see page 34)

2.2.46 gdxAcronymGetInfo

```
function gdxAcronymGetInfo(pgdx: pointer; N: Integer; var AName: ShortString; var Txt:
ShortString; var AIndx: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymGetInfo (see page 34)

2.2.47 gdxAcronymGetMapping

```
function gdxAcronymGetMapping(pgdx: pointer; N: Integer; var orgIndx: Integer; var newIndx:
Integer; var autoIndex: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymGetMapping (see page 35)

2.2.48 gdxAcronymIndex

```
function gdxAcronymIndex(pgdx: pointer; V: Double): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymIndex (see page 35)

2.2.49 gdxAcronymName

```
function gdxAcronymName(pgdx: pointer; V: Double; var AName: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymName (see page 35)

2.2.50 gdxAcronymNextNr

```
function gdxAcronymNextNr(pgdx: pointer; NV: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymNextNr (see page 36)

2.2.51 gdxAcronymSetInfo

```
function gdxAcronymSetInfo(pgdx: pointer; N: Integer; const AName: ShortString; const Txt:
ShortString; AIndx: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymSetInfo (see page 36)

2.2.52 gdxAcronymValue

```
function gdxAcronymValue(pgdx: pointer; AIndx: Integer): Double; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAcronymValue (see page 36)

2.2.53 gdxAddAlias

```
function gdxAddAlias(pgdx: pointer; const Id1: ShortString; const Id2: ShortString): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAddAlias (see page 37)

2.2.54 gdxAddSetText

```
function gdxAddSetText(pgdx: pointer; const Txt: ShortString; var TxtNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAddSetText (see page 37)

2.2.55 gdxAutoConvert

```
function gdxAutoConvert(pgdx: pointer; NV: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxAutoConvert (see page 37)

2.2.56 gdxClose

```
function gdxClose(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxClose (see page 38)

2.2.57 gdxCreate

Calls gdxGetReady (see page 88) to load the library and creates a gdx object. The library is loaded from OS default location. The name for the library is automatic.

```
function gdxCreate(var Ap: pointer; var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`var Ap: pointer`

On return contains a pointer to a gdx object or nil when loading the library failed

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxCreateX (⧉ see page 79), gdxCreateD (⧉ see page 78), gdxCreateL (⧉ see page 78)

2.2.58 gdxCreateD

Calls gdxGetReadyD (⧉ see page 88) to load the library and creates a gdx object. Load the library from from a specified directory. The name for the library is automatic.

```
function gdxCreateD(var Ap: pointer; const Dir: ShortString; var Msg: shortString): boolean;
```

Unit

gdxAPIfuncs (⧉ see gdxAPIfuncs.pas, page 100)

Parameters

`var Ap: pointer`

On return contains a pointer to a gdx object or nil when loading the library failed

`const Dir: ShortString`

Directory to load library from.

`var Msg: shortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxCreate (⧉ see page 77), gdxCreateX (⧉ see page 79), gdxCreateL (⧉ see page 78)

2.2.59 gdxCreateL

Calls gdxGetReadyL (⧉ see page 88) to load the library and creates a gdx object. Load library from full path specified; no changes are made to the name (platform and file extension)

```
function gdxCreateL(var Ap: pointer; const LibName: ShortString; var Msg: shortString):
boolean;
```

Unit

gdxAPIfuncs (⧉ see gdxAPIfuncs.pas, page 100)

Parameters

`var Ap: pointer`

On return contains a pointer to a gdx object or nil when loading the library failed

`const LibName: ShortString`

Full path of the library.

`var Msg: shortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxCreate (see page 77), gdxCreateX (see page 79), gdxCreateD (see page 78)

2.2.60 gdxCreateX

Calls gdxGetReadyX (see page 89) to load the library and creates a gdx object. Tries to load the library from main program directory; if that fails, loads library from the OS default location. The name for the library is automatic.

```
function gdxCreateX(vap Ap: pointer; var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`vap Ap: pointer`

On return contains a pointer to a gdx object or nil when loading the library failed

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxCreate (see page 77), gdxCreateD (see page 78), gdxCreateL (see page 78)

2.2.61 gdxCurrentDim

```
function gdxCurrentDim(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxCurrentDim (see page 38)

2.2.62 gdxDataErrorCount

```
function gdxDataErrorCount(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataErrorCount (see page 38)

2.2.63 gdxDataErrorRecord

```
function gdxDataErrorRecord(pgdx: pointer; RecNr: Integer; var KeyInt: TgdxUELIndex; var
Values: TgdxValues): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataErrorRecord (see page 38)

2.2.64 gdxDataReadDone

```
function gdxDataReadDone(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadDone (see page 39)

2.2.65 gdxDataReadFilteredStart

```
function gdxDataReadFilteredStart(pgdx: pointer; SyNr: Integer; const FilterAction:
TgdxUELIndex; var NrRecs: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadFilteredStart (see page 39)

2.2.66 gdxDataReadMap

```
function gdxDataReadMap(pgdx: pointer; RecNr: Integer; var KeyInt: TgdxUELIndex; var Values:
TgdxValues; var DimFrst: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadMap (see page 40)

2.2.67 gdxDataReadMapStart

```
function gdxDataReadMapStart(pgdx: pointer; SyNr: Integer; var NrRecs: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadMapStart (see page 40)

2.2.68 gdxDataReadRaw

```
function gdxDataReadRaw(pgdx: pointer; var KeyInt: TgdxUELIndex; var Values: TgdxValues; var
DimFrst: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadRaw (see page 41)

2.2.69 gdxDataReadRawFast

```
function gdxDataReadRawFast(pgdx: pointer; SyNr: Integer; DP: TDataStoreProc; var NrRecs:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadRawFast (see page 41)

2.2.70 gdxDataReadRawFastFilt

```
function gdxDataReadRawFastFilt(pgdx: pointer; SyNr: Integer; const UelFilterStr:
TgdxStrIndex; DP: TDataStoreFiltProc): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadRawFastFilt (see page 41)

2.2.71 gdxDataReadRawStart

```
function gdxDataReadRawStart(pgdx: pointer; SyNr: Integer; var NrRecs: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadRawStart (see page 42)

2.2.72 gdxDataReadSlice

```
function gdxDataReadSlice(pgdx: pointer; const UelFilterStr: TgdxStrIndex; var Dimen: Integer;
DP: TDataStoreProc): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadSlice (see page 42)

2.2.73 gdxDataReadSliceStart

```
function gdxDataReadSliceStart(pgdx: pointer; SyNr: Integer; var ElemCounts: TgdxUELIndex):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadSliceStart (see page 43)

2.2.74 gdxDataReadStr

```
function gdxDataReadStr(pgdx: pointer; var KeyStr: TgdxStrIndex; var Values: TgdxValues; var
DimFrst: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadStr (see page 43)

2.2.75 gdxDataReadStrStart

```
function gdxDataReadStrStart(pgdx: pointer; SyNr: Integer; var NrRecs: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataReadStrStart (see page 44)

2.2.76 gdxDataSliceUELS

```
function gdxDataSliceUELS(pgdx: pointer; const SliceKeyInt: TgdxUELIndex; var KeyStr:
TgdxStrIndex): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataSliceUELS (see page 44)

2.2.77 gdxDataWriteDone

```
function gdxDataWriteDone(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteDone (see page 45)

2.2.78 gdxDataWriteMap

```
function gdxDataWriteMap(pgdx: pointer; const KeyInt: TgdxUELIndex; const Values: TgdxValues):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteMap (see page 45)

2.2.79 gdxDataWriteMapStart

```
function gdxDataWriteMapStart(pgdx: pointer; const SyId: ShortString; const ExplTxt:
ShortString; Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteMapStart (see page 45)

2.2.80 gdxDataWriteRaw

```
function gdxDataWriteRaw(pgdx: pointer; const KeyInt: TgdxUELIndex; const Values: TgdxValues):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteRaw (see page 46)

2.2.81 gdxDataWriteRawStart

```
function gdxDataWriteRawStart(pgdx: pointer; const SyId: ShortString; const ExplTxt:
ShortString; Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteRawStart (see page 46)

2.2.82 gdxDataWriteStr

```
function gdxDataWriteStr(pgdx: pointer; const KeyStr: TgdxStrIndex; const Values: TgdxValues):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteStr (see page 47)

2.2.83 gdxDataWriteStrStart

```
function gdxDataWriteStrStart(pgdx: pointer; const SyId: ShortString; const ExplTxt:
ShortString; Dimen: Integer; Typ: Integer; UserInfo: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxDataWriteStrStart (see page 47)

2.2.84 gdxErrorCount

```
function gdxErrorCount(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxErrorCount (see page 47)

2.2.85 gdxErrorStr

```
function gdxErrorStr(pgdx: pointer; ErrNr: Integer; var ErrMsg: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxErrorStr (see page 48)

2.2.86 gdxFileInfo

```
function gdxFileInfo(pgdx: pointer; var FileVer: Integer; var ComprLev: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFileInfo (see page 48)

2.2.87 gdxFileVersion

```
function gdxFileVersion(pgdx: pointer; var FileStr: ShortString; var ProduceStr: ShortString):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFileVersion (see page 48)

2.2.88 gdxFilterExists

```
function gdxFilterExists(pgdx: pointer; FilterNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFilterExists (see page 49)

2.2.89 gdxFilterRegister

```
function gdxFilterRegister(pgdx: pointer; UelMap: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFilterRegister (see page 49)

2.2.90 gdxFilterRegisterDone

```
function gdxFilterRegisterDone(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFilterRegisterDone (see page 49)

2.2.91 gdxFilterRegisterStart

```
function gdxFilterRegisterStart(pgdx: pointer; FilterNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFilterRegisterStart (see page 49)

2.2.92 gdxFindSymbol

```
function gdxFindSymbol(pgdx: pointer; const SyId: ShortString; var SyNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxFindSymbol (see page 50)

2.2.93 gdxFree

Finish any pending write operations by calling gdxClose (see page 77) and frees the object

```
procedure gdxFree(var Ap: pointer);
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`var Ap: pointer`

Pointer to gdx object; will be set to nil.

2.2.94 gdxGetDLLVersion

```
function gdxGetDLLVersion(pgdx: pointer; var V: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetDLLVersion (see page 50)

2.2.95 gdxGetDomainElements

```
function gdxGetDomainElements(pgdx: pointer; SyNr: Integer; DimPos: Integer; FilterNr:
Integer; DP: TDomainIndexProc; var NrElem: Integer; Uptr: Pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetDomainElements (see page 50)

2.2.96 gdxGetElemText

```
function gdxGetElemText(pgdx: pointer; TxtNr: Integer; var Txt: ShortString; var Node:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetElemText (see page 51)

2.2.97 gdxGetLastError

```
function gdxGetLastError(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetLastError (see page 52)

2.2.98 gdxGetMemoryUsed

```
function gdxGetMemoryUsed(pgdx: pointer): Int64; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetMemoryUsed (see page 52)

2.2.99 gdxGetReady

Load the library from OS default location. The name for the library is automatic.

```
function gdxGetReady(var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

2.2.100 gdxGetReadyD

Load the library from from a specified directory. The name for the library is automatic.

```
function gdxGetReadyD(const Dir: ShortString; var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`const Dir: ShortString`

Directory to load library from.

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxGetReady (see page 88), gdxGetReadyX (see page 89), gdxGetReadyL (see page 88)

2.2.101 gdxGetReadyL

Load library from full path specified; no changes are made to the name (platform and file extension)

```
function gdxGetReadyL(const LibName: ShortString; var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`const LibName: ShortString`

Full path of the library.

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxGetReady (see page 88), gdxGetReadyX (see page 89), gdxGetReadyD (see page 88)

2.2.102 gdxGetReadyX

Tries to load the library from main program directory; if that fails, loads library from the OS default location. The name for the library is automatic.

```
function gdxGetReadyX(var Msg: ShortString): boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Parameters

`var Msg: ShortString`

Error message if library load failed; empty otherwise.

Return Value

True if library loaded successfully; False otherwise.

See Also

gdxGetReady (see page 88), gdxGetReadyD (see page 88), gdxGetReadyL (see page 88)

2.2.103 gdxGetSpecialValues

```
function gdxGetSpecialValues(pgdx: pointer; var AVals: TgdxSVals): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetSpecialValues (see page 52)

2.2.104 gdxGetUEL

```
function gdxGetUEL(pgdx: pointer; UelNr: Integer; var Uel: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxGetUEL (see page 52)

2.2.105 gdxLibraryLoaded

Returns true if the gdx library is loaded; false otherwise.

```
function gdxLibraryLoaded: boolean;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

2.2.106 gdxLibraryUnload

Unload the gdx library.

```
procedure gdxLibraryUnload;
```

Unit

gdxAPIfuncs (see gdxAPIfuncs.pas, page 100)

Notes

The gdxCreate (see page 77) functions and gdxFree (see page 86) count the number of live objects, and this procedure will raise an error if there are one or more live gdx objects.

2.2.107 gdxMapValue

```
function gdxMapValue(pgdx: pointer; D: Double; var sv: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxMapValue (see page 53)

2.2.108 gdxOpenAppend

```
function gdxOpenAppend(pgdx: pointer; const FileName: ShortString; const Producer:
ShortString; var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxOpenAppend (see page 53)

2.2.109 gdxOpenRead

```
function gdxOpenRead(pgdx: pointer; const FileName: ShortString; var ErrNr: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxOpenRead (see page 54)

2.2.110 gdxOpenWrite

```
function gdxOpenWrite(pgdx: pointer; const FileName: ShortString; const Producer: ShortString;
var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxOpenWrite (see page 54)

2.2.111 gdxOpenWriteEx

```
function gdxOpenWriteEx(pgdx: pointer; const FileName: ShortString; const Producer:
ShortString; Compr: Integer; var ErrNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxOpenWriteEx (see page 55)

2.2.112 gdxRenameUEL

```
function gdxRenameUEL(pgdx: pointer; const OldName: ShortString; const NewName: ShortString):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxRenameUEL (see page 56)

2.2.113 gdxResetSpecialValues

```
function gdxResetSpecialValues(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxResetSpecialValues (see page 56)

2.2.114 gdxSetHasText

```
function gdxSetHasText(pgdx: pointer; SyNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSetHasText (see page 56)

2.2.115 gdxSetReadSpecialValues

```
function gdxSetReadSpecialValues(pgdx: pointer; const AVals: TgdxSVals): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSetReadSpecialValues (see page 56)

2.2.116 gdxSetSpecialValues

```
function gdxSetSpecialValues(pgdx: pointer; const AVals: TgdxSVals): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSetSpecialValues (see page 57)

2.2.117 gdxSetTextNodeNr

```
function gdxSetTextNodeNr(pgdx: pointer; TxtNr: Integer; Node: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSetTextNodeNr (see page 57)

2.2.118 gdxSetTraceLevel

```
function gdxSetTraceLevel(pgdx: pointer; N: Integer; const s: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSetTraceLevel (see page 57)

2.2.119 gdxSymbIndxMaxLength

```
function gdxSymbIndxMaxLength(pgdx: pointer; SyNr: Integer; var LengthInfo: TgdxUELIndex):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbIndxMaxLength (see page 58)

2.2.120 gdxSymbMaxLength

```
function gdxSymbMaxLength(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbMaxLength (see page 58)

2.2.121 gdxSymbolAddComment

```
function gdxSymbolAddComment(pgdx: pointer; SyNr: Integer; const Txt: ShortString): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolAddComment (see page 58)

2.2.122 gdxSymbolDim

```
function gdxSymbolDim(pgdx: pointer; SyNr: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolDim (see page 58)

2.2.123 gdxSymbolGetComment

```
function gdxSymbolGetComment(pgdx: pointer; SyNr: Integer; N: Integer; var Txt: ShortString):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolGetComment (see page 59)

2.2.124 gdxSymbolGetDomain

```
function gdxSymbolGetDomain(pgdx: pointer; SyNr: Integer; var DomainSyNrs: TgdxUELIndex):
```

```
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolGetDomain (see page 59)

2.2.125 gdxSymbolGetDomainX

```
function gdxSymbolGetDomainX(pgdx: pointer; SyNr: Integer; var DomainIDs: TgdxStrIndex):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolGetDomainX (see page 59)

2.2.126 gdxSymbolInfo

```
function gdxSymbolInfo(pgdx: pointer; SyNr: Integer; var SyId: ShortString; var Dimen:
Integer; var Typ: Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolInfo (see page 60)

2.2.127 gdxSymbolInfoX

```
function gdxSymbolInfoX(pgdx: pointer; SyNr: Integer; var RecCnt: Integer; var UserInfo:
Integer; var ExplTxt: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolInfoX (see page 60)

2.2.128 gdxSymbolSetDomain

```
function gdxSymbolSetDomain(pgdx: pointer; const DomainIDs: TgdxStrIndex): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolSetDomain (see page 60)

2.2.129 gdxSymbolSetDomainX

```
function gdxSymbolSetDomainX(pgdx: pointer; SyNr: Integer; const DomainIDs: TgdxStrIndex):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSymbolSetDomainX (see page 61)

2.2.130 gdxSystemInfo

```
function gdxSystemInfo(pgdx: pointer; var SyCnt: Integer; var UelCnt: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxSystemInfo (see page 61)

2.2.131 gdxUELMaxLength

```
function gdxUELMaxLength(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELMaxLength (see page 61)

2.2.132 gdxUELRegisterDone

```
function gdxUELRegisterDone(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterDone (see page 61)

2.2.133 gdxUELRegisterMap

```
function gdxUELRegisterMap(pgdx: pointer; UMap: Integer; const Uel: ShortString): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterMap (see page 62)

2.2.134 gdxUELRegisterMapStart

```
function gdxUELRegisterMapStart(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterMapStart (see page 62)

2.2.135 gdxUELRegisterRaw

```
function gdxUELRegisterRaw(pgdx: pointer; const Uel: ShortString): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterRaw (see page 62)

2.2.136 gdxUELRegisterRawStart

```
function gdxUELRegisterRawStart(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterRawStart (see page 63)

2.2.137 gdxUELRegisterStr

```
function gdxUELRegisterStr(pgdx: pointer; const Uel: ShortString; var UelNr: Integer):
Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterStr (see page 63)

2.2.138 gdxUELRegisterStrStart

```
function gdxUELRegisterStrStart(pgdx: pointer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUELRegisterStrStart (see page 63)

2.2.139 gdxUMFindUEL

```
function gdxUMFindUEL(pgdx: pointer; const Uel: ShortString; var UelNr: Integer; var UelMap:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUMFindUEL (see page 63)

2.2.140 gdxUMUelGet

```
function gdxUMUelGet(pgdx: pointer; UelNr: Integer; var Uel: ShortString; var UelMap:
Integer): Integer; stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

`pgdx: pointer`

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUMUelGet (see page 64)

2.2.141 gdxUMUelInfo

```
function gdxUMUelInfo(pgdx: pointer; var UelCnt: Integer; var HighMap: Integer): Integer;
stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Parameters

```
pgdx: pointer
```

Pointer to GDX structure

Notes

This is the Delphi wrapped version of TGXFileObj.gdxUMUelInfo (see page 64)

2.2.142 gdxXFree

```
procedure gdxXFree(var pgdx: pointer); stdcall;
```

Unit

gdxdclib (see gdxdclib.dpr, page 100)

Description

comp returns the compatibility mode: 0: client is too old for the DLL, no compatibility 1: client version and DLL version are the same, full compatibility 2: client is older than DLL, but defined as compatible, backward compatibility 3: client is newer than DLL, forward compatibility

2.3 Structs and Records

These are all structs and records that are contained in this documentation.

2.3.1 uInt64

```
uInt64 = record
  case integer of
    1: (i: Int64;);
    2: (p: pointer;);
  end;
```

Unit

gxfile (see gxfile.pas, page 102)

Description

gdxcb2.inc end

2.4 Types

These are all types that are contained in this documentation.

2.4.1 PGXFile

```
PGXFile = pointer;
```

Unit

gxdefs (see gxdefs.pas, page 101)

Description

Pointer to a GDX data structure

2.4.2 TDomainIndexProc_F

```
TDomainIndexProc_F = procedure (var RawIndex: Integer; var MappedIndex: Integer; var Uptr:
Int64);
```

Unit

gxdefs (see gxdefs.pas, page 101)

Description

gdxcb1.inc end

2.4.3 TgdxStrIndex

```
TgdxStrIndex = gmsspecs.TStrIndex;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Array type for an index using strings

2.4.4 TgdxSVals

```
TgdxSVals = array[TgdxSpecialValue] of double;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Array type for passing special values

2.4.5 TgdxUELIndex

```
TgdxUELIndex = gmsspecs.TIndex ;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Array type for an index using integers

2.4.6 TgdxValues

```
TgdxValues = gmsspecs.tvarreca;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Array type for passing values

2.5 Variables

These are all variables that are contained in this documentation.

2.5.1 DLLLoadPath

```
DLLLoadPath: ShortString;
```

Unit

gxfile (↗ see gxfile.pas, page 102)

Description

can be set by loader, so the 'dll' knows where it is loaded from

2.5.2 local_DP

```
local_DP: TDomainIndexProc;
```

Unit

gdxdclib (↗ see gdxdclib.dpr, page 100)

Description

Entry points having string arguments

2.6 Constants

These are all constants that are contained in this documentation.

2.6.1 DOMC_EXPAND

```
DOMC_EXPAND = -1;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Indicator for a growing index position

2.6.2 DOMC_STRICT

```
DOMC_STRICT = 0;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Indicator for a mapped index position

2.6.3 DOMC_UNMAPPED

```
DOMC_UNMAPPED = -2;
```

Unit

gxdefs (↗ see gxdefs.pas, page 101)

Description

Indicator for an unmapped index position

2.6.4 ERR_OPEN_DOMSMARKER3

```
ERR_OPEN_DOMSMARKER3 = -100063;
```

Unit

gxfile (↗ see gxfile.pas, page 102)

Description

Errors from gdxcopy

2.7 gdxAPIfuncs.pas

Unit Overview

Functions in Unit gdxAPIfuncs

gdxCreate (↗ see page 77)
Calls gdxGetReady (↗ see page 88) to load the library and creates a gdx object. The library is loaded from OS default location. The name for the library is automatic.

gdxCreateD (↗ see page 78)
Calls gdxGetReadyD (↗ see page 88) to load the library and creates a gdx object. Load the library from from a specified directory. The name for the library is automatic.

gdxCreateL (↗ see page 78)
Calls gdxGetReadyL (↗ see page 88) to load the library and creates a gdx object. Load library from full path specified; no changes are made to the name (platform and file extension)

gdxCreateX (↗ see page 79)
Calls gdxGetReadyX (↗ see page 89) to load the library and creates a gdx object. Tries to load the library from main program directory; if that fails, loads library from the OS default location. The name for the library is automatic.

gdxGetReady (↗ see page 88)
Load the library from OS default location. The name for the library is automatic.

gdxGetReadyD (↗ see page 88)
Load the library from from a specified directory. The name for the library is automatic.

gdxGetReadyL (↗ see page 88)
Load library from full path specified; no changes are made to the name (platform and file extension)

gdxGetReadyX (↗ see page 89)
Tries to load the library from main program directory; if that fails, loads library from the OS default location. The name for the library is automatic.

gdxLibraryLoaded (↗ see page 89)
Returns true if the gdx library is loaded; false otherwise.

gdxLibraryUnload (↗ see page 89)
Unload the gdx library.

This units documents a few functions for using the GDX library

2.8 gdxdclib.dpr

Unit Overview

Functions in Unit gdxdclib

BgdxDataReadStr (↗ see page 64)
BgdxDataSliceUELS (↗ see page 65)
BgdxSymbolGetDomainX (↗ see page 65)
CgdxAcronymAdd (↗ see page 65)
CgdxAcronymGetInfo (↗ see page 65)
CgdxAcronymName (↗ see page 65)
CgdxAcronymSetInfo (↗ see page 66)
CgdxAddAlias (↗ see page 66)
CgdxAddSetText (↗ see page 66)
CgdxDataReadRawFastFilt (↗ see page 66)
CgdxDataReadSlice (↗ see page 67)
CgdxDataReadStr (↗ see page 67)
CgdxDataSliceUELS (↗ see page 67)
CgdxDataWriteMapStart (↗ see page 67)

CgdxDataWriteRawStart (see page 68)	CgdxDataWriteStr (see page 68)
CgdxDataWriteStrStart (see page 68)	CgdxErrorStr (see page 68)
CgdxFileVersion (see page 69)	CgdxFindSymbol (see page 69)
CgdxGetDLLVersion (see page 69)	CgdxGetElemText (see page 69)
CgdxGetUEL (see page 69)	CgdxOpenAppend (see page 70)
CgdxOpenRead (see page 70)	CgdxOpenWrite (see page 70)
CgdxOpenWriteEx (see page 70)	CgdxRenameUEL (see page 71)
CgdxSetTraceLevel (see page 71)	CgdxSymbolAddComment (see page 71)
CgdxSymbolGetComment (see page 71)	CgdxSymbolGetDomainX (see page 72)
CgdxSymbolInfo (see page 72)	CgdxSymbolInfoX (see page 72)
CgdxSymbolSetDomain (see page 72)	CgdxSymbolSetDomainX (see page 73)
CgdxUELRegisterMap (see page 73)	CgdxUELRegisterRaw (see page 73)
CgdxUELRegisterStr (see page 73)	CgdxUMFindUEL (see page 73)
CgdxUMUelGet (see page 74)	FgdxDataReadRawFastFilt (see page 74)
FgdxGetDomainElements (see page 74)	gdxAcronymAdd (see page 74)
gdxAcronymCount (see page 75)	gdxAcronymGetInfo (see page 75)
gdxAcronymGetMapping (see page 75)	gdxAcronymIndex (see page 75)
gdxAcronymName (see page 76)	gdxAcronymNextNr (see page 76)
gdxAcronymSetInfo (see page 76)	gdxAcronymValue (see page 76)
gdxAddAlias (see page 76)	gdxAddSetText (see page 77)
gdxAutoConvert (see page 77)	gdxClose (see page 77)
gdxCurrentDim (see page 79)	gdxDataErrorCount (see page 79)
gdxDataErrorRecord (see page 79)	gdxDataReadDone (see page 80)
gdxDataReadFilteredStart (see page 80)	gdxDataReadMap (see page 80)
gdxDataReadMapStart (see page 80)	gdxDataReadRaw (see page 81)
gdxDataReadRawFast (see page 81)	gdxDataReadRawFastFilt (see page 81)
gdxDataReadRawStart (see page 81)	gdxDataReadSlice (see page 81)
gdxDataReadSliceStart (see page 82)	gdxDataReadStr (see page 82)
gdxDataReadStrStart (see page 82)	gdxDataSliceUELS (see page 82)
gdxDataWriteDone (see page 83)	gdxDataWriteMap (see page 83)
gdxDataWriteMapStart (see page 83)	gdxDataWriteRaw (see page 83)
gdxDataWriteRawStart (see page 84)	gdxDataWriteStr (see page 84)
gdxDataWriteStrStart (see page 84)	gdxErrorCount (see page 84)
gdxErrorStr (see page 85)	gdxFileInfo (see page 85)
gdxFileVersion (see page 85)	gdxFilterExists (see page 85)
gdxFilterRegister (see page 85)	gdxFilterRegisterDone (see page 86)
gdxFilterRegisterStart (see page 86)	gdxFindSymbol (see page 86)
gdxFree (see page 86) *Finish any pending write operations by calling gdxClose (see page 77) and frees the object*	gdxGetDLLVersion (see page 87)
gdxGetDomainElements (see page 87)	gdxGetElemText (see page 87)
gdxGetLastError (see page 87)	gdxGetMemoryUsed (see page 87)
gdxGetSpecialValues (see page 89)	gdxGetUEL (see page 89)
gdxMapValue (see page 90)	gdxOpenAppend (see page 90)
gdxOpenRead (see page 90)	gdxOpenWrite (see page 90)
gdxOpenWriteEx (see page 91)	gdxRenameUEL (see page 91)
gdxResetSpecialValues (see page 91)	gdxSetHasText (see page 91)
gdxSetReadSpecialValues (see page 91)	gdxSetSpecialValues (see page 92)
gdxSetTextNodeNr (see page 92)	gdxSetTraceLevel (see page 92)
gdxSymbIndxMaxLength (see page 92)	gdxSymbMaxLength (see page 93)
gdxSymbolAddComment (see page 93)	gdxSymbolDim (see page 93)
gdxSymbolGetComment (see page 93)	gdxSymbolGetDomain (see page 93)
gdxSymbolGetDomainX (see page 94)	gdxSymbolInfo (see page 94)
gdxSymbolInfoX (see page 94)	gdxSymbolSetDomain (see page 94)
gdxSymbolSetDomainX (see page 95)	gdxSystemInfo (see page 95)
gdxUELMaxLength (see page 95)	gdxUELRegisterDone (see page 95)
gdxUELRegisterMap (see page 96)	gdxUELRegisterMapStart (see page 96)
gdxUELRegisterRaw (see page 96)	gdxUELRegisterRawStart (see page 96)
gdxUELRegisterStr (see page 96)	gdxUELRegisterStrStart (see page 97)
gdxUMFindUEL (see page 97)	gdxUMUelGet (see page 97)
gdxUMUelInfo (see page 97)	gdxXFree (see page 98)

Variables in Unit gdxdclib

local_DP (see page 99)

Delphi Library program generated by apiwrapper for GAMS Version 24.4.0

2.9 gxdefs.pas

Unit Overview

Types in Unit gxdefs

PGXFile (see page 98)	TDomainIndexProc_F (see page 98)
TgdxStrIndex (see page 98)	TgdxSVals (see page 99)
TgdxUELIndex (see page 99)	TgdxValues (see page 99)

Constants in Unit gxdefs

DOMC_EXPAND (see page 99)	DOMC_STRICT (see page 100)
DOMC_UNMAPPED (see page 100)	

used by gxfile.pas (see page 102) and any program needing the constants and types for using the gdxio.dll

2.10 gxfile.pas

Unit Overview

Classes in Unit gxfile

TGXFileObj (see page 32)	uInt64 (see page 98)

Variables in Unit gxfile

DLLLoadPath (see page 99)

Constants in Unit gxfile

ERR_OPEN_DOMSMARKER3 (see page 100)

This unit defines the GDX Object as a Delphi object. This unit is used by GDXIO.DPR which is used to build the GDXIO DLL called gdxdclib.dll in Windows

Index