

---

# Table of Contents

<b>Part I Overview</b>	<b>2</b>
<b>Part II Options</b>	<b>3</b>
<b>Part III Examples</b>	<b>5</b>
1 Model gdx2access1: import trnsport.gdx .....	5
2 Model gdx2access2: import setttext .....	5
3 Model gdx2access3: import a large table .....	6
4 Model gdx2access4: special value mapping .....	6
5 Model gdx2access5: renaming fields .....	7
<b>Part IV References</b>	<b>8</b>
<b>Index</b>	<b>0</b>

---

# 1 Overview

GDX2ACCESS is a tool to dump the contents of a GDX file to an MS Access file (.mdb or .accdb file). Every identifier gets its own table in the database. For instance when we save the results of the transport model from the model library:

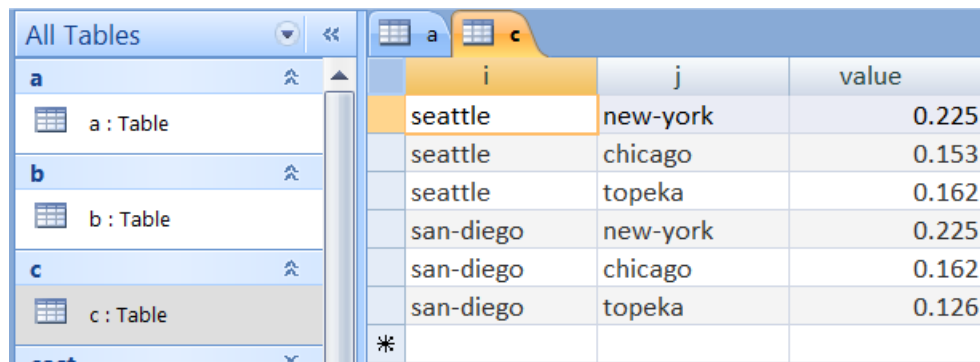
```
C:\GAMS support\settext>gamslib trnsport
Copy ASCII: trnsport.gms

C:\GAMS support\settext>gams trnsport gdx=trnsport lo=2

C:\GAMS support\settext>gdxdump trnsport.gdx symbols
* GDX dump of trnsport.gdx
* Library in use : C:\GAMS23.3
* Library version: GDX Library      Nov  1, 2009 23.3.3 WIN 14596.15043 VIS x86/MS Windows
* File version   : GDX Library      Nov  1, 2009 23.3.3 WIN 14596.15043 VIS x86/MS Windows
* Producer       : GAMS Base Module Nov  1, 2009 23.3.3 WIN 14929.15043 VIS x86/MS Windows
* File format    : 7
* Compression    : 0
* Symbols        : 12
* Unique Elements: 5
  Symbol Dim Type Explanatory text
  1 a          1 Par capacity of plant i in cases
  2 b          1 Par demand at market j in cases
  3 c          2 Par transport cost in thousands of dollars per case
  4 cost        0 Equ define objective function
  5 d          2 Par distance in thousands of miles
  6 demand      1 Equ satisfy demand at market j
  7 f          0 Par freight in dollars per case per thousand miles
  8 i          1 Set canning plants
  9 j          1 Set markets
10 supply      1 Equ observe supply limit at plant i
11 x          2 Var shipment quantities in cases
12 z          0 Var total transportation costs in thousands of dollars
```

The example shows how we copy the trnsport.gms model from the model library, and then solve it. The option gdx=filename will save the complete symbol table to a GDX file. The option lo=2 tells GAMS to save the log to a file (in this case trnsport.log) instead of writing it to the screen. The gdxdump will display the contents of the GDX file (the option symbols will only display the table of contents, rather than all data). Once we have a GDX file we can use GDX2ACCESS to create an .MDB or a .ACcdb file. Versions of MicroSoft Office prior to version 2007 use the file extension .mdb; version 2007 use the file extension .accdb.

```
C:\GAMS support\settext>gdx2access trnsport.gdx
GDX Access      ALFA 23Mar10 23.4.0 WIN 16693.16738 VS8 x86/MS Windows
Creating C:\GAMS support\settext\templ.accdb with Access: 0.48 seconds
Using temp directory C:\Users\Paul\AppData\Local\Temp\
  i.  Insert: 0.00 seconds
  j.  Insert: 0.00 seconds
  a.  Insert: 0.00 seconds
  b.  Insert: 0.00 seconds
  d.  Dump: 0.00 seconds Load: 0.05 seconds
  f.  Insert: 0.00 seconds
  c.  Dump: 0.00 seconds Load: 0.05 seconds
  x.  Dump: 0.00 seconds Load: 0.03 seconds
  z.  Insert: 0.00 seconds
  cost. Insert: 0.00 seconds
  supply. Insert: 0.02 seconds
  demand. Insert: 0.02 seconds
Renaming C:\GAMS support\settext\templ.accdb -> trnsport.accdb
Total elapsed time: 0.92 seconds
```

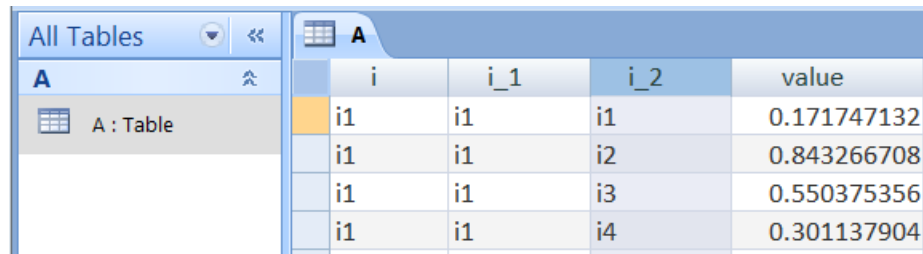


i	j	value
seattle	new-york	0.225
seattle	chicago	0.153
seattle	topeka	0.162
san-diego	new-york	0.225
san-diego	chicago	0.162
san-diego	topeka	0.126

Figure 1: trnsport.gdx converted to an Access database

The resulting MDB, opened with MS Access is shown in figure 1. The complete process shown here can be automated as is shown in section 3.1. As can be seen, every identifier is stored in its own table. If no domain information is available for a symbol, index positions get a column with labels dim1, dim2, etc. If domain information is available, the columns will use that information but keeping the names unique. The small example below shows how the index positions are made unique using the (relaxed) domain information.

```
set i /i1*i5/;
parameter A(i, i, i);
Alias(i, j, k);
A(i, j, k) = uniform(0,1);
execute_unload 'ex1.gdx', A;
execute '=gdx2access ex1.gdx';
```



i	i_1	i_2	value
i1	i1	i1	0.171747132
i1	i1	i2	0.843266708
i1	i1	i3	0.550375356
i1	i1	i4	0.301137904

Figure 2: Symbol A shown with unique column names

For parameters, the value is stored in a column named `value`, while variables and equations have columns `level`, `marginal`, `lowerbound`, `upperbound`. A possible additional field (`scale` for NLP's, `priority` for MIP's, `stage` for stochastic problems) is not exported. If needed you can assign such a quantity to a parameter before writing the GDX file. For an example see figure 2.

## 2 Options

Options are specified in an .INI file. By default, the file `gdx2access.ini` located in the same directory as `gdx2access.exe` is consulted. If this file is not available, the program will continue using default settings. It is also possible to tell the program to use a different .ini file. This is done by using an extra argument of the form `@inifile`. An example would be:

```
gdx2access myfile.gdx @myinifile.ini
```

In this case the program will not read `gdx2access.ini` located in the same directory as `gdx2access.exe` but rather `myinifile.ini` in the current directory.

The ini file can contain two sections: [settings] and [debug]. A complete ini file with all possible settings looks like:

```
[settings]
smdir=c:\tmp
inf=1.0e100
mininf=-1.0e100
eps=0.0
na=0.0
undf=0.0
scalartable=1
etflag=1
[debug]
method=5
thresholdcount=5
keepfiles=1
```

Description for the settings and debug sections:

[settings]	default	meaning
smdir	<windowstemp>	Directory for temporary scratch files.
inf	1.0e100	The value used for GAMS INF
mininf	-1.0e100	The value used for GAMS -INF
eps	0.0	The value used for GAMS EPS
na	0.0	The value used for GAMS NA
undf	0.0	The value used for GAMS UNDF
scalartable	0	Value 0/1; when set to 1, combine scalars of the same type in a single table. The names for these tables are fixed: ScalarParameter, ScalarEquation and ScalarVariable
etflag	0	Value 0/1; When set to 1, include the text strings for sets containing element texts.
dbversion	0	Create database using current default format
		9: Create a database in the Microsoft Access 2000 (.mdb) file format.
		10: Create a database in the Microsoft Access 2002-2003 (.mdb) file format.
		12: Create a database in the Microsoft Office Access 2007 (.accdb) file format.

[debug]	default	meaning
method	5	Select algorithm to insert data into the Access database
		1: Write a CSV file and use the TransferText action to read this into Access. This is fast, but does not always work when non-US language settings are used.
		2: Use recordset.add to add records. This is slow, but does not use intermediate files.
		3: Write a tab delimited file with a complete file specification (schema.ini) and use the ISAM Text driver to import the data. This is fast and should work in international settings.
		4: For small data use method=2 and for larger data items use method=1.
		5: For small data use method=2 and for larger data items use method=3.
thresholdcount	5	When to change between algorithms when using method=4 or method=5. The default is 5 records.
keepfiles	0	Value 0/1; when set to 1, the program will not delete intermediate scratch files.

## 3 Examples

[Model gdx2access1: import trnsport.gdx](#)  
[Model gdx2access2: import setttext](#)  
[Model gdx2access3: import a large table](#)  
[Model gdx2access4: special value mapping](#)  
[Model gdx2access5: renaming fields](#)

### 3.1 Model gdx2access1: import trnsport.gdx

This example will solve the trnsport.gms model from the model library and generate a GDX file containing the complete symbol table. This GDX file is exported to Access and MS Access is launched to inspect the results. This is a small example that should run very quickly.

```
$ontext
Test of GDX2ACCESS. Dumps all symbols of
trnsport.gms to trnsport.mdb
$offtext
execute '=gamslib trnsport';
execute '=gams trnsport lo=3 gdx=trnsport';
execute '=gdx2access trnsport.gdx';
execute '=ShellExecute trnsport.mdb';
```

Notes: the equal signs in from of the external programs indicate we don't go through a shell (e.g. command.com or cmd.exe). This will improve reliability in case the external program is not found. In such a case a proper error will be triggered. Without the '=' such errors go undetected and the GAMS model will continue.

The command ShellExecute will launch Access to view the .MDB file, see [5]. This assumes that the version of of Access installed is a version prior to version 2007. Later versions will generate a database with the extension .accdb and the ShellExecute call needs to be changed as follows:

```
* view generated file create
execute '=ShellExecute trnsport.accdb';
```

### 3.2 Model gdx2access2: import setttext

In this example we write a few sets to a GDX file; two of the sets written have explanatory text for set elements. We use an option (etflag) to get this text saved in the Access database. Without using the option, only the set tuples are saved in the database.

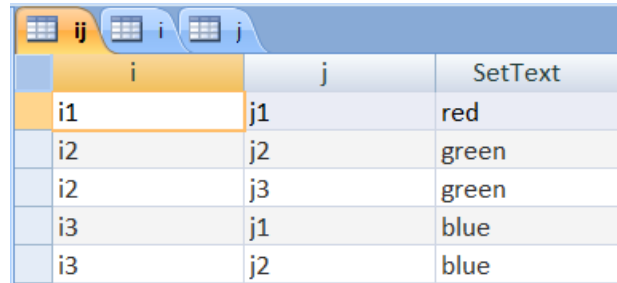
Running this example on a machine with Access 2007 or a later version installed, will create a database with the .accdb file extension that cannot be read by an older version of Access. We use an option (dbversion) to save the database in the .mdb format.

```
set i /i1 one, i2 two, i3 three, i4 four/
    j /j1*j4/
    ij(i,j) / i1.j1 red, i2.(j2,j3) green, i3.(j1,j2) blue /;
$gdxout example2.gdx
$unload i j ij
$gdxout

$onecho > task2.ini
[settings]
etflag=1
dbversion=10
$offecho
```

```
$call =gdx2access example2.gdx @task2.ini
```

When we open the database and inspect the table create from the set `ij` we see the explanatory texts stored:



i	j	SetText
i1	j1	red
i2	j2	green
i2	j3	green
i3	j1	blue
i3	j2	blue

Set explanatory text shown

### 3.3 Model gdx2access3: import a large table

This is an artificial example where we generate a large identifier in GAMS: a parameter with a million elements.

```
$ontext
Test of GDX2ACCESS. Dumps a large symbol (a million elements) to an Access Database.
$offtext
set i /i1*i1000/;
alias (i,j);
parameter p(i,j);
p(i,j) = uniform(-100,100);
execute_unload 'test.gdx',p;
execute 'gdx2access test.gdx';
```

### 3.4 Model gdx2access4: special value mapping

To store special values like INF, EPS, NA in a numeric field in the database, GDX2Access uses a mapping. This mapping can be changed using an INI file.

```
$ontext
Test of GDX2ACCESS.
Check special value mapping.
$offtext
$onecho > m.ini
[settings]
inf=1
mininf=2
eps=3
na=4
undf=5
$offecho
parameter p(*) /
i1 inf
i2 -inf
i3 eps
i4 na
/;
p('i5') = 1/0;
display p;
*
* save parameter p in p.mdb, as table p
* special values are translated to default values:
* INF -> 1.0e100
```

```
* -INF -> -1.0e100
* EPS,NA,UNDF -> 0
*
execute_unload "p.gdx",p;
execute '=gdx2access p.gdx @m.ini';
```

We cannot enter a value of Undef directly so we use a division by zero to get the Undef value.

When we view the generated GDX file in the GAMS IDE, the special values are shown:

i1	+Inf
i2	-Inf
i3	Eps
i4	NA
i5	Undef

Symbol P  
as viewed  
in the IDE

Viewing the resulting table in Access shows how the mapping for special values was applied:

p	
dim1	value
i1	1
i2	2
i3	3
i4	4
i5	5

Symbol P in Access

### 3.5 Model gdx2access5: renaming fields

GDX2Access will use names like i, j, dim1, dim2, Value etc. In some cases this may not be convenient, e.g. when more descriptive field names are required. In the following model we will show how a small script in VBscript[3, 2] can handle this task. The script will rename the columns i, j, and Value in table c to ifrom, jto, and transportcost.

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

parameters
  a(i) "capacity of plant i in cases" /
    seattle 350
    san-diego 600/
  b(j) "demand at market j in cases" /
    new-york 325
    chicago 300
    topeka 275 /;

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 ;
*
* export to gdx file.
execute_unload "c.gdx",c;

* move to access database
* column names are i and j
*
execute "=gdx2access c.gdx";
*
* rename columns
*
execute "=cscript access.vbs";
*
* view results
*
execute "=shellexecute c.accdb";

$onecho > access.vbs
'this is a VBscript script
WScript.Echo "Running script: access.vbs"
set oa = CreateObject("Access.Application")
set oDAO = oa.DBEngine
Wscript.Echo "DAO Version: " & oDAO.version
Set oDB = oDAO.openDatabase("%system.fp%c.accdb")
Wscript.Echo "Opened : " & oDB.name
Set oTable = oDB.TableDefs.Item("c")
Wscript.Echo "Table : " & oTable.name
' rename fields
oTable.Fields.Item("i").name = "ifrom"
oTable.Fields.Item("j").name = "jto"
oTable.Fields.Item("Value").name = "transportcost"
Wscript.Echo "Renamed fields"
oDB.Close
Wscript.Echo "Done"
$offecho
```

## 4 References

1. Microsoft Corp., GetTempPath, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/gettemppath.asp>, March 2005.
  2. \_\_\_\_\_, VBScript Language Reference, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbscripttoc.asp>, 2005.
  3. \_\_\_\_\_, VBScript User's Guide, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbstutor.asp>, 2005.
-