Test Data Generation Script

The test data generation script is a language that allows the user to define complete test data generation project with connection options, output options, database or file schema definition and test data generation rules.

There are two ways to create the script: export* it from DTM Data Generator or create it manually.
* - see export limitations.

There are three main reasons to deal with scripts and script compiler:

- The compiled file works 3 to 10 times faster than the same project executed in DTM Data Generator.
- The generated executable files can be redistributed without additional or hidden fees.
- Use generated C# output as a part of another project.

The script is a text file with one command per line. This document describes all commands and provides a few examples.

Syntax notes:

- The compiler ignores spaces and tabs outside quoted values.
- Any property must be quoted by ' " ' sign.
- Comments begin with '#'. The compiler ignores text starting this sign to end of the line.

The DTM Data Generation Script Compiler is a tool that converts script to Windows executable file and (optional) C# source file. The licensed compiler user allowed to redistribute created executable file without royalty.

The compiler converts data generation script to windows executable and, optional C# source code file.

Requirements: the compiler and produced executable file require .net 2.0 or newer. Suitable operating systems are Windows 2000 and newer.

Command line parameters

Parameter	Default Value	Description
-s<file>	Mandatory parameter	Name with full or relative path to source script (.DGL).
-e<file>	Mandatory parameter	Name with full or relative path to executable file (.EXE).
-c<file>	No C# will be created by default	Name with full or relative path to C# intermediate code (.cs).
-l<file>	Script compiler uses console as default output	Name with full or relative path for custom compiler log file.
-o<file>	Program uses console as default output	Name with full or relative path for created program output file.
-r<0|1>	1 (means 'YES')	When the switch is '1' the compiler will copy supplemental DLL add 'Value Library' to directory with generated EXE file.
-h		Show brief help for command line parameters.

Note: command line switches are case sensitive.

Examples

```
dgl_compiler.exe -sD:\SOURCES\NowrthWind.dgl -eD:\EXECUTABLES\NowrthWind.exe
```

DTM Data Generation Script Compiler Errors

This document describes script compiler error codes and messages

Code	Message	Description
ERR1000	Compiler expects at least two command line parameters: script file name and output file name.	The program found 0 or 1 arguments.
ERR1001	Invalid command line parameter format...	Parameter has no ' -' or '/' prefix or too short.
ERR1002	Invalid command line parameter 's' format: no script file provided.	-s without script name provided.
ERR1003	Invalid command line parameter 'e' format: no output file name provided.	-e without file name provided.
ERR1004	Invalid command line parameter 'c' format: no C# file name provided.	-c without .CS file name.
ERR1005	Invalid command line parameter 'r' format.	The program expects -r0 or -r1.
ERR1006	Unknown command line parameter...	The parameter is not recognized.
ERR1007	Both source and output files must be specified.	Source script or output file name empty.
ERR1008	Invalid command line parameter 'l' format: no log file name provided.	-l without log file name.
ERR1009	Invalid command line parameter 'o' format: no output file name provided.	-o without log file name.
ERR2000	Script file '...' does not exists or inaccessible.	The program could not access to provided source file.
ERR2001	Error reading source script file...	Some IO problem happens during source file reading.
ERR2002	Error writing C# file...	Some IO problem happens during C# file writing.
ERR2003	Could not switch to custom error log '...'	The program could not open or create custom log file specified by -l command line parameter.
ERR2004	Could not copy supplemental objects to target directory: '...'	The program could copy DLL or value library to target folder due to IO error.
ERR3000	Incorrect 'rule' command	The ' Rule' command has incorrect format.
ERR3001	Unknown rule type '...'	Rule type is not in acceptable list
ERR3002	Wrong ' Field' command	Field command has invalid number of parameters.
ERR3003	Wrong ' Pattern' command for 'Field'	Pattern command has invalid number of parameters.
ERR3004	Connection property without value '...'	Some connection property command has no parameter or has extra parameters.
ERR3005	Project property without value	Some project property command has no

	'...'	parameter or has extra parameters.
ERR3005	Wrong 'rowcount' command.	The Rowcount command expects at least two parameters.
ERR3006	Wrong 'rowcount:records' command	The Rowcount-Records command expects exact 3 parameters (Rowcount, 'Records', Number)
ERR3007	Rule property without value '...'	Some rule property command has no parameter or has extra parameters.
ERR3008	Incorrect 'Table' command	The table command expects exact 3 arguments (Table, 'Name' and table name)
ERR3009	Unrecognized command '...'	The command is not recognized.
ERR3010	Command '...' is not suitable in this context '...'	The command in wrong position. For example, connection property in the field definition.
ERR3011	Rule property '...' is not recognized.	Provided property name not recognized as known rule property.
ERR3012	Project property '...' is not recognized.	Provided property name not recognized as known project property.
ERR3013	Connection property '...' is not recognized.	Provided property name not recognized as known connection property.
ERR3014	Wrong 'rowcount:table' command.	The Rowcount-Table command expects exact 5 parameters (Rowcount, 'table', name, 'column', name)
ERR3015	Wrong 'rowcount:sql' command.	The Rowcount-SQL command expects exact 3 parameters (Rowcount, 'sql', statement)
ERR3016	Wrong 'rowcount:random' command.	The Rowcount-Random command expects exact 3 parameters (Rowcount, 'random', number1:number2)
ERR3017	Wrong 'rowcount:driven' command.	The Rowcount-Driven command expects exact 7 parameters (Rowcount, 'driven', table, 'column', name, 'random',number1:number2)
ERR3018	Wrong 'rowcount:random' range '...' provided.	The expects range as "number1:number2" value.
ERR3019	Wrong 'rowcount:driven' range '...' provided.	The expects range as "number1:number2" value.
ERR3020	Incorrect named generator definition.	Incorrect number of properties or no 'pattern' property provided.
ERR3021	Incorrect variable definition.	Incorrect number of properties for 'Variable' command.
ERR3022	Incorrect or incomplete variable definition.	'type' or 'def' property not found in the variable definition.
ERR3023	'Name' command is suitable for 'Object' and 'Table' rules only.	'Name' can't be applied to rule type except 'Object' and 'Table'.
ERR3024	Incorrect or incomplete 'Name' command for 'Object' and 'Table' rule.	The compiler expects 2 properties ('Name' and template).
ERR3025	Incorrect or incomplete 'Object' command.	'Object' command requires 5 properties: 'object', 'type', type, 'sql', statements.
ERR3026	Transaction size is not positive integer value.	Not a number or negative value provided as transaction size.
ERR3027	'ObjectsN' counter is not positive integer value.	Not a number or negative value provided as object counter.
ERR3028	'Object' command without 'Text' property.	No mandatory 'Text' property found in the object definition.

ERR3029	'Object' command without 'Type' property.	No mandatory 'Type' property found in the object definition.
ERR3030	The rowcount:records property is not positive integer.	The compiler expects positive integer value as 'Records' property.
ERR4000	C# compiler error '...'	The C# compiler could not compile generated .CS file.
ERR4001	Unclassified script compiler error: '...'	Unclassified or unhandled problem. Please contact support team.

To redistribute the compiled script the user should copy following files to target location:

- <EXE file generated by script compiler>
- dgl_runtime.dll
- gentmpl.dll
- common3base.dll
- library.mdb (optional, required if $Lib or $LibGroup function used in the script).
- libaccess.dll (optional, required if $Lib or $LibGroup function used in the script).

End User License Agreement

This License Agreement covers all existing versions of DTM Data Generation Script Compiler: demo and commercial. License Agreement is a legal agreement between the end-user and DTM soft. By installing or otherwise using DTM Data Generation Script Compiler, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of License Agreement, please do not install or use DTM Data Generation Script Compiler.

General Information

1. DTM soft is exclusive owner of all DTM Data Generation Script Compiler copyrights. The product is licensed, not sold. DTM Data Generation Script Compiler is protected by copyright laws and international copyright treaties.
2. Demo version. Anyone may install and use demo version of DTM Data Generation Script Compiler for evaluation and testing purposes.
3. Commercial version. You may install and use one copy of DTM Data Generation Script Compiler on a single computer. The primary user of the computer on which DTM Data Generation Script Compiler is installed may make a second copy for his or her exclusive use on a portable computer.
4. You may not reverse engineer, modify, decompile, or disassemble DTM Data Generation Script Compiler. The Software Product is licensed as a single product. Its component parts may not be separated for use on more than one computer.
5. You may not rent, lease, or lend DTM Data Generation Script Compiler. Also, You may not resell, or otherwise transfer for value, DTM Data Generation Script Compiler.
6. Without prejudice to any other rights, DTM soft may terminate this License Agreement if you fail to comply with the terms and conditions of this Agreement. In such event, you must destroy all copies of DTM Data Generation Script Compiler and all of its component parts.
7. You may permanently transfer all of your rights under this license, provided you retain no copies, you transfer all of DTM Data Generation Script Compiler (including all component parts), and the recipient agrees to the terms of this license.
8. DTM Data Generation Script Compiler IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE DTM Data Generation Script Compiler AT YOUR OWN RISK. DTM soft WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Redistribution Policy

1. You may create and run any copes of executables generated by the script compiler free of charge.
2. You may distribute generated executables without additional charges including outside your company except cases mentioned in the item #5.
3. Nobody (you or your end users) is allowed to reverse engineer, modify, decompile, or disassemble generated executables or supplemental materials (DLL, library, etc).
4. You may not alter any copyright, trademark or patent notice in the generated executables or supplemental materials.
5. You may not include the distributable components in malicious, deceptive or unlawful programs.

Support and Upgrades

During one year after ordering any license except "Site" and "World" licenses, you are entitled to free technical services and support for DTM Data Generation Script Compiler which is provided to you by DTM soft. During this period, e-mail support is unlimited and includes technical and support questions. Also, during one year, you may access to free updates to DTM Data Generation Script Compiler when and as DTM soft publishes them on www.sqledit.com. After end of the described period you may continue to use the software product in accordance with the terms of this Agreement except free support and upgrades. After end of the free support and updates period (one year), you may purchase annual Upgrade and Support subscription. If you ordered a few DTM licenses, you will access to free upgrade and support period and will use subscriptions independently.

Trademarks information

DTM Data Generation Script Compiler is trademark of DTM soft.

Export script from DTM Data Generator Limitations

- Only Professional and Enterprise editions of the tool support export to script option.
- The database connection information export feature supports ODBC with DSN connections only.
- The generator does not support cross-database references in the export feature.

Connection Block Header

The "Connection" command starts the block of commands that defines database connection information. The block can contain following commands: connection type, user, password, owner, database, DSN.

Syntax:
Connection

Note: connection definition is required even "No Exec" project property is 'true'. The pattern engine requires active connection in any case.

Example:

```
Connection
 Type ="ODBC"
 DSN ="NorthWind"
 User ="sa"
 Password ="sa -pwd"
 Owner ="dbo"
 Database ="NorthwindOriginal"
```

Connection Type or Database Interface

The connection type defines database interface. Currently the script compiler supports "ODBC" connection type only.

Syntax:
Type="<type or interface>"

Connection: User name or Login

The "User" command allows to specify user name or login name for database connection.

Syntax:
User="<user name or login name>"

Connection: Password

The "Password" command allows to specify optional password for database connection.

Syntax:
Password="<password as clear text>"

Connection: Schema or Owner name

The "Owner" command is a way to define schema name (or owner name depends on database system).

Syntax:
Owner="<schema name or owner>"

Connection: Database Name

The "Database" command allows to define database name. The option is depend on database system.

Syntax:
Database="<database>"

Connection: Data Source Name

The DSN option of the connection section is a way to provide ODBC Data Source Name for connection.

Syntax:
DSN="<data source name>"

Project Properties

The project section consist of a few project properties and rule definitions.

There are:

- No Exec
- To File
- Truncate SQL
- Statement Delimiter
- Text Delimiter
- Text File Header
- Quote Always
- Project Description
- Author
- Prologue SQL Script
- Epilogue SQL Script

Example:

```
Project   Name="NorthwindCode3"
 NoExec ="1"
 ToFile ="d:\1.sql"
 TruncateSQL ="1"
 StmtDelimiter =""
 TxtDelimiter ="<tab>"
 TxtHeader ="1"
 TxtQuoteAlways ="0"
 sqlBefore ="--  prologue"
 sqlAfter ="--  epilogue"
 note="The  northwind  project"
 author ="Martin"
```

'No Execute' option

If the "no Execute" option is TRUE (value "1") the script does not modify database directly but creates output files only. This property is optional, default value is FALSE ("0").

Syntax:
NoExec="<1|0>"

'To File' option

The "to file" option defines output file for SQL script. This value is optional. In case the file name does not specified, the script will generate no SQL file as output.

Syntax:
ToFile="<file name with path>"

'Truncate SQL File' option

This option instructs to truncate output SQL file before each project execution. It is default behavior. Otherwise (FALSE value, "0") the script will append rows to existing file.

Syntax:
TruncateSQL="<1|0>"

'Truncate Text File' option

This option instructs to truncate text output file before each project execution. It is default behavior. Otherwise (FALSE value, "0") the script will append rows to existing file.

Syntax:
TruncateTXT="<1|0>"

'To Text File' option

The "to text file" option defines output file for text output. This value is optional. In case the file name does not specified, the script will generate no text file as output.

Syntax:
ToTxtFile="<file name with path>"

'To XML File' option

The "to XML file" option defines output file for XML output. This value is optional. In case the file name does not specified, the script will generate no XML file as output.

Syntax:
ToXMLFile="<file name with path>"

'Truncate XML File' option

This option instructs to truncate output XML file before each project execution. It is default behavior. Otherwise (FALSE value, "0") the script will append rows to existing file.

Syntax:
TruncateXML="<1|0>"

'SQL Statement Delimiter' option

This option allows the user to specify custom SQL statement delimiter. The property is optional. By default, the script will use no separator between generated statements.

Syntax:
StmtDelimiter="<delimiter>"

'Text file delimiter' option

This option allows to define value-to-value separator for text output. By default, it is <tab> sign.

Syntax:
TxtDelimiter="<delimiter>"

'Text file header' option

If this option is switched off ("0" value) the script will skip header row with column names. Default value is 1.

Syntax:
TxtHeader="<1|0>"

'Text quote always' option

If this option is switched on the program quotes by '"' sign each value in the output text file.

Syntax:
TxtQuoteAlways="<1|0>"

The '1' value means 'yes' when '0' corresponds to 'no'. Default value is 0.

'Prologue SQL' Project Option

The prologue script is a set of SQL statements that will be executed before data generation rules. This script is optional.

Syntax:
sqlBefore="<SQL statements>"

'Epilogue SQL' Project Option

The epilogue script is a set of SQL statements that will be executed after data generation rules. This script is optional.

Syntax:
sqlBefore="<SQL statements>"

Project Description Option

The user allowed to add to script a short description of the project. The script compiler does not use this field. It is for information purpose only.

Syntax:
Note="<project description>"

Project Author Option

The user allowed to add an project's author name to the script. The script compiler does not use this field. It is for information purpose only.

Syntax:
Author="<author>"

Named Generators Section

The Named Generators Section is a header of the named generator list. It has no options.

Syntax:
NamedGenerators

Example:

```
NamedGenerators
Name="COMPANY"    Pattern="$Lib(Companies,,40,0)"
```

Named Generator Definition

This command defines one Named Generator. It has tow mandatory parameters: name and pattern.

Syntax:
Name="<named generator name>" Pattern="<test data generation pattern>"

Variables Section

The "Variables" command starts list fo variables definition section. There is no parameters.

Syntax:
Variables

Example:

```
Variables
 Name="#A50"   Type="Constant"   Def="50"
```

Variable Definition

The "Variable" definition command has three mandatory properties: variable name, type and definition.

Syntax:
Name="<name of the variable>" Type="constant|query" Def="<constant or script>"

Rule Definition

The "rule" command is a header of the test data generation Rule definition. The only mandatory parameter is rule type.

Syntax:
Rule="data|file|clear|tables|objects"

Clear Rule

The Clear Rule definition contains a list of tables to be cleared.

Example:

```
Rule  Type="clear"
 Table   Name="dbo.Order  Details"
 Table    Name="dbo.Products"
 Table    Name="dbo.Orders"
```

Object Name Option

The table name defines rule-related table to be populated or cleared depends on rule type.

Syntax:
Table Name="<table name>"

Note: quotation is not required for this command

Data Rule

The Data Rule definition has a few properties and list of columns of the table to be populated or scrambled.
There are properties:

- Row Count
- Table Name
- Insertion Mode
- Transaction Size
- Fields

Example:

```
Rule  Type="data"
 Table     Name="dbo.CustomerDemographics"
 Rowcount   Records="25"
 TransactionSize ="500"
 InsMode ="replace"
 Fields
  Field    Name="CustomerTypeID"
   DataType  Code="-8"  Name="nchar"  length="10"
   Pattern ="$Unique($RString(1,10,5,0,0))"
  Field    Name="CustomerDesc"
   DataType  Code="-10"  Name="ntext"
   Pattern ="$IfR(10,\NULL,$RString(1,4096,5,0,0))"
```

Row Count Option

This command defines number of rows to be generated. It has a five forms.

Syntax form #1:
Rowcount Records="<Number of records to be generated>"
Example:
Rowcount Records="25000"

Syntax form #2:
Rowcount Random="<Number from:Number to>"
Example:
Rowcount Random="10:100"

Syntax form #3:
Rowcount Table="<Table Name>" Column="<Column Name>"
Example:
Rowcount Table="Order" Column="OrderID"

Syntax form #4:
Rowcount Sql="<SQL statement>"
Example:
Rowcount sql="select count(*) from Order where OrderDate>'12.01.2000'"

Syntax form #5:
Rowcount Driven="<Table Name>" Column="<Column Name>" Random="<Number from:Number to>"
Example:
Rowcount Driven="Order" Column="OrderID" Random="10:100"

Transaction Size Option

The "Transaction Size" option defines number of statements in the transaction. This value is performance related and has no influence to generated data. This rule property is optional, 500 statements is default value.

Syntax:
TransactionSize=<size>

Rule Mode Option

This command defines data insertion mode for the rule.

Syntax:
InsMode="append|replace|update|scramble"

'Prologue SQL' Rule Option

The "sqlBefore" command defines rule-level SQL script. The script will be executed before the rule.

Syntax:
sqlBefore="<SQL statements>"

'Epilogue SQL' Rule Option

The "sqlAfter" command defines rule-level SQL script. It will be executed after the rule.

Syntax:
sqlAfter="<SQL statements>"

Rule-Level SQL Output File Option

The "RuleToFile" defines custom SQL script for rule output. It overrides 'ToFile' option for the single rule.

Syntax:
RuleToFile="<file name with path>"

Rule-Level Text Output File Option

The "RuleToTxtFile" defines custom text output file for rule output. It overrides 'ToTxtFile' option for the single rule.

Syntax:
RuleToTxtFile="<file name with path>"

Rule-Level XML Output File Option

The "RuleToXMLFile" defines custom XML output file for rule output. It overrides 'ToXmlFile' option for the single rule.

Syntax:
RuleToXMLFile="<file name with path>"

Fields Section

The "Fields" command starts the list of the fields of the table associated with some rule. It has no parameters.

Syntax:
Fields

Example:

```
Fields
 Field   Name="CustomerTypeID"
  DataType  Code="-8"  Name="nchar"  length="10"
  Pattern ="$Unique($RString(1,10,5,0,0))"
 Field   Name="CustomerDesc"
  DataType  Code="-10"  Name="ntext"
  Pattern ="$IfR(10,\NULL,$RString(1,4096,5,0,0))"
```

One Filed Definition

The Field definition consists of "field" command and depended type and pattern commands.

Syntax:
Field Name="<field name>"

Filed Data Type

The "DataType" command describes data type of the field.

Syntax:
DataType Code="<data type code>" Name="<data type name>" [Length="<length>"] [Null="0|1"]

The '0' value means 'no null' when '1' means 'null allowed'.

The acceptable data type codes are:

- -7 is 'bit'
- -6 is 'tinyint'
- -5 is 'bigint' ('int8' for PostgreSQL)
- -4 is 'longvarbinary' ('image' for SQL server or 'blob' for Oracle, Interbase and DB2)
- -3 is 'varbinary'
- -2 id 'binary'
- -1 is 'lognvarchar' ('text' for SQL Server and PostgreSQL, 'clob' for Oracle and DB2)
- 1 is 'char'
- 2 is 'numeric'
- 3 is 'decimal' ('number' for Oracle)
- 4 is 'integer'
- 5 is 'smallint' ('int2' for PostgreSQL)
- 6 is 'float' ('float4' for PostgreSQL)
- 7 is 'real'
- 8 is 'double' ('double precision' for Oracle, 'float8' for PostgreSQL)
- 9 is 'datetime' or 'date'
- 10 is 'time'
- 11 is 'timestamp'
- 12 is 'varchar' ('varchar2' for Oracle)

Please contact our support team if your data type is not in this list.

Data Generation Pattern for the Filed

The "Pattern" command defines test data generation pattern for the field. It is pattern engine compatible string.

Syntax:
Pattern="<pattern text>"

Tables Rule

The Tables rule generates a set of tables with same structure. The rule definition consists of following commands:

- Object Name Pattern
- Transaction Size
- Objects counter
- Row counter, how many rows should be inserted into each created table
- Fields definition section

Example:

```
Rule  Type="table"
 Name="A*"
 Rowcount   Records="10"
 TransactionSize ="500"
 ObjectsN ="10"
 Fields
    Field  Name="COL1"
      DataType  Code="1"  Name="char"  length="10"  Null="1"
        Pattern="$IfR(20,\NULL,$RString(,,5,0,0))"
    Field  Name="COL2"
      DataType  Code="4"  Name="int"  Null="1"
       Pattern="$Unique($Rint(,,))"
```

Object Name

The "Name" command defines a pattern for objects (table, view, procedure, etc) to be generated. It should contain '*' and/or '?' wildcards.

Syntax:
Name="<pattern>"

Example:

```
    Name="BB*"
```

Objects Counter

The "ObjectsN" command defines number of objects to be generated in Objects or Tables rule.

Syntax:
ObjectsN="<number of objects>"

Objects Rule

The Tables rule generates a set of database objects like procedures or triggers with same structure. The rule definition consists of following commands:

- Object Name Pattern
- Transaction Size
- Objects counter
- Object definition

Example:

```
Rule  Type="object"
 Name="BB*"
 TransactionSize ="500"
 ObjectsN="75"
 Object   Type="0"  Text="select  getdate()"
```

Object Definition command

This command defines database to be generated in the "objects" rule.

Syntax:
Object Type="<type code>" Text="<SQL statements>"

The applicable type codes are:

- 0: view
- 1: procedure
- 2: rule
- 3: role
- 4: trigger
- 5: index

Project Header

The "Project" command is a root item of the script. Only one "Project" command is allowed per script file and it must be first command. The optional parameter is "Name". The script compiler does not use this parameter now.

Syntax:
Project [Name="<project name>"]