Data Generation SDK

DTM Data Generation SDK (Software Developers Kit) is a set of functions that allows the programmer or advanced user to:

- automate DTM Data Generator operations (creation and modification of the projects)
- use DTM Data Generator together with customer's application
- add powerful data generation features to user's application, service or script

This SDK offers a few levels of services (API, Application programming interface):

- Basic or Low-Level API allows the user to generate single data items like strings, numbers or dates with specified properties.
- Fill Methods level API offers same fill methods as DTM Data Generator except "By Formula".
- Rule level API helps the user to operate with data generation rules. In most cases, the rule corresponds to one table to be populated. This level contains functions to manipulate individual data generation properties for each column of the table as well.
- Project level API operates with sets of rules. Also, this level supports project execution and results handling.

There are three editions of the SDK (Core, Standard, Professional):

Feature	Core	Standard	Professional
Basic Level support	Yes	Yes	Yes
Fill Methods level support	Yes*	Yes	Yes
Rule level support	No	Yes	Yes
Project Level support	No	Yes	Yes
x64 edition	No	No	Yes
Redistributable. The license allows royalty-free distribution of SDK-based solutions.	No	No	Yes

* - Excepts methods require database connections like "by SQL statement" or "From Table".

File	Folder	Description	Platform
dgsdk.dll	/	Main DLL with all API functions	Win32, Unicode, x86
dgsdk.lib	/	Import library for dgsdk.dll	Win32, Unicode, x86
dgsdk.dll	x64	Main DLL with all API functions	Unicode, x64
dgsdk.lib	x64	Import library for dgsdk.dll	Unicode, x64
common.dll	/	DLL with supplemental functions (not included in Core edition of SDK)	Win32, x86, ANSI
common.dll	x64	DLL with supplemental functions	x64, Unicode
dgsdk.chm	/	SDK documentation, Windows HTML help format	
library.mdb	/	Value Library	
reg_com_server.cmd	/	Command file for manual COM-object registration	
unreg_com_server.cmd	/	Command file for manual COM-object deregistration	
lowLevelAPI.c	examples.c	C example for low level API	
MaskMethod.c	examples.c	C example for by mask fill method	
PatternMethod.c	examples.c	C example for by pattern fill method	
ListMethod.c	examples.c	C example for from list fill method	
FileMethod.c	examples.c	C example for from file fill method	
IncrementalMethod.c	examples.c	C example for incremental fill method	
LibraryMethod.c	examples.c	C example for from library fill method	
TableMethod.c	examples.c	C example for from table fill method	
SQLMethod.c	examples.c	C example for by SQL statement fill method	
projectAPI.c	examples.c	C example for Project level API	
run_project.c	examples.c	C example for project execution function	
lowLevelAPI.cs	examples.cs	C# example for low level API	
MaskMethod.cs	examples.cs	C# example for by mask fill method	
SQLMethod.cs	examples.cs	C# example for by SQL statement fill method	
TableMethod.cs	examples.cs	C# example for from table fill method	
FileMethod.cs	examples.cs	C# example for from file fill method	
lowLevelAPI.asp	examples.asp	ASP (VBScript) example for low level API	
MaskMethod.asp	examples.asp	ASP (VBScript) example for by mask fill method	
LibraryMethod.asp	examples.asp	ASP (VBScript) example for from library fill method	
run_project.py	samples.py	Python example for project execution function	
projectAPI.py	samples.py	Python example for project API functions	

SQLMethod.py	samples.py	Python example for SQL fill method	
TableMethod.py	samples.py	Python example for Table fill method	
IncrementalMethod.py	samples.py	Python example for incremental fill method	
PatternMethod.py	samples.py	Python example for pattern fill method	
LibraryMethod.py	samples.py	Python example for Library fill method	
FileMethod.py	samples.py	Python example for File fill method	
lowLevelAPI.py	samples.py	Python example for low level API	

Note: distribution package can also contain a few supplemental or optional files not mentioned in the list.

General Functions Map

Function	API Level	SDK Edition
GetError	General	All
ConnectDB	General	STD, PRO
ExecDB	General	STD, PRO
DisconnectDB	General	STD, PRO
CloseH	General	All
ShortRandom	Low Level	All
IntRandom	Low Level	All
CharRandom	Low Level	All
CharRandomUpper	Low Level	All
CharRandomLower	Low Level	All
CharRandomDigit	Low Level	All
DoubleRandom	Low Level	All
StringRandom	Low Level	All
DateRandom	Low Level	All
TimeRandom	Low Level	All
InitList	Fill methods	All*
ListValue	Fill methods	All*
InitFile	Fill methods	All
FileValue	Fill methods	All
InitTable	Fill methods	STD, PRO
TableValue	Fill methods	STD, PRO
InitLibrary	Fill methods	All
LibraryValue	Fill methods	All
InitIncremental	Fill methods	All
IncrementalValue	Fill methods	All
InitMask	Fill methods	All
MaskValue	Fill methods	All
InitPattern	Fill methods	All
PatternValue	Fill methods	All
PatternEndOfRow	Fill methods	All
InitStatement	Fill methods	STD, PRO
StatementValue	Fill methods	STD, PRO
PrevValue	Fill methods	All
RunProject	Project	All
LoadProject	Project	All
SaveProject	Project	All
NewProject	Project	All
CloseProject	Project	All
nRules	Project	All
GetProjectProperty	Project	All
SetProjectProperty	Project	All

RemoveRule	Project	All
GetRule	Project	All
SetRule	Project	All
AddRule	Project	All
CloseRule	Rule	All
nRuleItems	Rule	All
GetRuleProperty	Rule	All
SetRuleProperty	Rule	All
GetRuleItem	Rule	All
SetRuleItem	Rule	All
AddRuleItem	Rule	All
RemoveRuleItem	Rule	All
GetItemProperty	Rule	All
SetItemProperty	Rule	All
CloseItem	Rule	All

* - COM object does not provide these functions.

Basic level API

Basic level is a low level set of functions. Each basic function generates and returns a single value.

- Short Integer Random value
- Integer Random value
- Random Symbol
- Double Random value
- Random String
- Random Date value
- Random Time value

See also: C/C++ example.

Short Random generator

This function generates short integer random value in range 0 to 32767. The user can change bound of the range using function parameters.

C/C++/C#

```
int ShortRandom(int from, int to)
```

Integer Random generator

This function generates integer random value in range 0 to 2147483647. The user can change bounds of the range using function parameters.

C/C++/C#
int IntRandom(int from, int to)

CharRandom, CharRandomUpper, CharRandomLower, CharRandomDigit
Returns random char from A-Z, a-z or 0-9 sets.
C/C++/C#
```
char CharRandom()
char CharRandomUpper() /* A-Z */
char CharRandomLower() /* a-z */
char CharRandomDigit() /* 0-9 */
```

Double Random generator

The function generates double precision value in specified range.

C/C++/C#

```
double DoubleRandom(double from, double to, int digits)
```

Random String

Creates a string with the specified length.
C/C++
char* StringRandom(int length)

C#
string StringRandom(int length)

Notes:

- maximum string length is 2048. You should call the function a few times to generate longer strings.
- if 'length' parameter <=0 the function returns empty string.

Random Date

Creates a string presentation for the random date in specified range with required format.
C/C++
char* DateRandom(char *format,char *from,char *to)

C#
string DateRandom(string format,string from,string to)

The format string can contain:
YY - two digits year value.
YYYY - four digits year value.
MM - month value from 1 to 12.
mon - month name (jan to dec).
DD - day from 1 to 31.
separators like '.','/' or ' -'.

Examples: DD.MM.YYYY, DD-mon-YYYY
Important: 'from' and 'to' should be compatible with the format specified by 'format' parameter.

Random Time

Creates a string presentation for the random time in specified range with required format.
C/C++
char* TimeRandom(char *format,char *from,char *to)

C#
string TimeRandom(string format,string from,string to)

The format string can contain:
HH - hours from 0 to 23 or from 0 to 12.
MM - minutes from 0 to 59.
SS - seconds from 0 to 59.
PM - PM or AM sign.

Examples: HH:MM:SS, HH:MM PM
Important: 'from' and 'to' should be compatible with the format specified by 'format'.

```c
#include "../dgsdkdefs.h"
#include <stdio.h>

int main()
{
  int i;
   printf("Line\tShort\tInteger\tSymbol\tUpper\tLower\tDigit\tDouble\tDate\tTime\tString\n");
  for(i=0; i<20; i++)
  {
     printf("%d\t%d\t%d\t%c\t%c\t%c\t%c\t%.2f\t%s\t%s\t%s\n",i+1,
        ShortRandom(100,200),           /* random short integer value from 100 to 200 */
        IntRandom(1000000,5000000),     /* random integer value from 1000000 to 5000000 */
        CharRandom(),                   /* random symbol A-Za-z0-9 */
        CharRandomUpper(),              /* random symbol A-Z */
        CharRandomLower(),              /* random symbol a-z */
        CharRandomDigit(),              /* random symbol 0-9 */
        DoubleRandom(100,100000,2),     /* random real value from 100 to 10000 */
        DateRandom("DD.MM.YYYY","01.01.2000","31.12.2009"),      /* random date */
        TimeRandom("HH:MM:SS","00:00:00","23:59:59"),            /* random time */
        StringRandom(10));                                       /* random string, length is 10 */
  }
}
```

Sample output:

Line	Short	Integer	Symbol	Upper	Lower	Digit	Double	Date	Time	String
1	184	2687879	c	A	a	7	61477.15	11.06.2004	17:47:34	bsGymPAqGG
2	151	4873365	m	G	h	4	90009.06	15.06.2005	06:18:22	z9u3QYbLWK
3	185	4305344	O	H	n	2	54915.19	12.08.2002	01:27:01	ZZBC1zb3T0
4	128	2683329	X	F	g	7	30329.34	25.03.2003	03:36:51	OI1vn6orjO
5	167	3931924	b	O	g	8	54411.26	27.03.2005	04:22:21	Dp7ndCzZ48
6	170	3087507	i	B	g	4	79659.77	19.10.2002	23:06:11	Jpx7h73yZK
7	165	2081689	g	O	t	5	96495.25	25.12.2001	11:19:55	BKnbngehki
8	194	2930965	x	E	h	7	88428.78	15.10.2001	02:33:33	N7lCIEdKWf
9	192	3070343	4	Y	b	3	64165.09	15.11.2007	05:28:47	8upXbpcCFQ
10	126	3120769	5	P	k	2	32535.43	22.02.2008	21:19:23	Tho0oNP7xb
11	113	2541147	C	C	x	8	95630.24	13.01.2006	04:35:50	wNgUFm7ucw
12	192	3489601	Z	Q	w	2	31657.90	29.03.2005	16:02:04	59SfZkFz2Z
13	135	4459713	u	S	v	5	9425.37	21.07.2001	02:09:10	wpEmebGWRk
14	100	2021164	c	E	j	3	21706.71	24.01.2004	09:08:09	8usyxsPIvY
15	162	2318600	6	S	l	3	10413.20	27.12.2008	12:20:46	saag9dsNQy
16	139	1462316	7	A	c	1	39767.33	14.06.2008	00:39:26	Aa4XR8fukK
17	159	3265841	J	P	y	4	53047.80	12.10.2005	14:42:09	iatwEvS65s
18	165	1910200	X	D	i	2	9448.53	01.10.2006	07:18:24	W50nYLXevN
19	188	3068114	O	F	o	0	50466.28	06.09.2002	17:06:41	DAaDdEAww7
20	182	4133063	M	U	q	2	81574.14	04.10.2002	06:09:17	juZ5QSncWz

Fill methods level API

There are fill methods:

- From list
- From text file
- From table
- From Values Library
- Incremental
- By Mask (obsolete)
- By Pattern
- By SQL statement

All fill methods consist of two functions: initialization and data generation. The first function saves fill methods properties into internal structures and prepares handle (DG_HANDLE type). The second (data generation) function accepts this handle as parameter.

Important: before use you should copy to local buffer result of any function of this API level.

All functions of this group return generated value as string. The user should made required conversions manually.

SDK offers function **PrevValue** that returns previously generated value for specified method's handle. This function is helpful for depended values generation.
C/C++
char *PrevValue(DG_HANDLE handle)
C#
string PrevValue(int handle)

See also: handles.

List Initialization

C/C++

DG_HANDLE InitList(int values, char *list[])
values - number of values the list;
list - array of strings for list's content.

Generate value

C/C++

char *ListValue(DG_HANDLE handle, int sequentially)

C#

string ListValue(int handle, int sequentially)
Not 0 'sequentially' value means the function will return values from the list sequentially, random otherwise.

Example

C/C++

```
#include   "../dgsdkdefs.h"
#include  <stdio.h>

int  main()
{
        int  i;
        char   *list[]   =   {"a10","a20","a30","a40","a50","a60","a70","a80","a90","a100",
                              "b10","b20","b30","b40","b50","b60","b70","b80","b90","b100"};
        char   buf1[5],buf2[5];
        DG_HANDLE        handle   =   InitList(20,list);

        if(handle==0)
                printf("Coild  not  allocate  Handle  for  Fill  Method\n");
        else
        {
                printf("Line\tValue   Random\tValue   Sequential\n");
                for(i=0;  i<20;  i++)
                {
                        strcpy(buf1,ListValue(handle,0));
                        strcpy(buf2,ListValue(handle,1));
                        printf("%d\t%s\t%s\n",i+1,buf1,buf2);
                }
                CloseH(handle);
        }
}
```

List Initialization
C/C++
DG_HANDLE InitFile(char *FileName)

C#/VBA/scripts
int InitFile(string FileName)
'FileName' parameter should contain full or relative path to file with values.

Generate value
C/C++
char* FileValue(DG_HANDLE handle, int sequentially)
C#/VBS/scripts
string FileValue(int handle, int sequentially)
Not 0 'sequentially' value means the function will return values from the file sequentially, random otherwise.

From table fill method

This fill method allows the user to generate data based on database table's column.

Table Initialization
C/C++
DG_HANDLE InitTable(DG_HANDLE connection, char *table, char *column)

C#/VBA/scripts
int InitTable(int connection, string table, string column)
'connection' is a handle returned by ConnectDB function.
'table' - table name
'column' - column name
Important: you should pass quoted string for tables like "Order Details" to this function. Please use native quotation symbols of your database system.

Generate value
C/C++
char* TableValue(DG_HANDLE handle, int sequentially)

C#/VBA/scripts
string TableValue(int handle, int sequentially)
Not 0 'sequentially' value means the function will return values from the table sequentially, random otherwise.

Note: this fill method not supported by core SDK version.

Library Initialization
C/C++
DG_HANDLE InitLibrary(char *LibraryLocation, char *LibraryItem)

C#/VBA/scripts
int InitLibrary(string LibraryLocation, string LibraryItem)
'LibraryLocation' is a full or relative path to the file contains Values Library.
'LibraryItem' specifies list of values. See table below for acceptable options.

Generate value
C/C++
char* LibraryValue(DG_HANDLE handle,int sequentially)

C#/VBA/scripts
string LibraryValue(int handle,int sequentially)
Not 0 'sequentially' value means the function will return values from the library item sequentially, random otherwise.

Default Values library contains at least following items:

Item Name	Description
Colors	English colors (145 items)
Names	Sample full names (about 380 items)
Occupations	
LastNames	List of sample last names
departments	Typical departments
Industries	List of industries
CurrencyCodes	World currency codes (3 symbols)
monthes	English months list
companies	The list of about 680 most known companies
FemaleFirstNames	Female names
MaleFirstNames	Male names
EuropeanCities	List of European cities
EuropeanCountries	List of European countries
EuropeanCapitals	List of capitals of European countries
Countries	World countries list
region	World regions
US_ZipCodes	ZIP codes (USA)
US_States	List of US states
US_StateCapitals	List of US state's capitals

Note: all DTM Data Generator customers can create own library or modify default library with Library Builder and included sources.

Incremental method Initialization
C/C++/C#
DG_HANDLE InitIncremental(int Start, int Step, int UseEach, int CycleLength)
'Start' is a first value of the sequence.
'Step' will be added at each iteration.
'UseEach' means how many times generated value will be used before add 'Step' value.
'CycleLength' defines after what number of iterations current value will be reinitialized by 'Start' value. 0 means no cycle.

Generate value
C/C++
char* IncrementalValue(DG_HANDLE handle)

C#/VBA/scripts
string IncrementalValue(int handle)

Mask Initialization
C/C++
DG_HANDLE InitMask(char *mask)

C#
int InitMask(string mask)

'mask' is definition of the mask that will be used to generate value.

Generate value
C/C++
char* MaskValue(DG_HANDLE handle)

C#
string MaskValue(int handle)

Masks

The mask is a text string with any number of 'A', 'a' and 'N' characters. During the value generation process the 'A' and 'a' characters is replaced with a random letter (from 'A' to 'Z' and from 'a' to 'z') and the 'N' character is replaced with a random figure. {n} means iterate last sign 1 to n times. All the rest of the mask characters will be moved to the result value without any changes.
If you want to specify several different masks for filling a field, you should use the "From the List" method and specify masks as values with the "Use list items as a mask" mode on.
You can use '\' character for escape next mask sign. For example, \a will be used as a 'a' letter without any replacements. Also, you can use I or I{n} item for autoincremental field. n is field size, I{3} means 001,002,...,999

There are mask items:

- A - letter from 'A' to 'Z'.
- a - letter from 'a' to 'z'.
- N - digit from '0' to '9'.
- Z - hexadecimal digit from '0' to '9' and 'A' to 'F'.
- {n} - repeater, value will be used from 1 to n times. n should be from 1 to 999.
- {=n} - repeater, value will be used n times.
- {n;m} - repeater, value will be used from n to m times. n should be less then m and both numbers from 1 to 999.
- I{n} - autoincremental value from 1 to 10**n.
- # - copy value of the last used {} block.

Compatibility warning: this fill method is obsolete. Please use it for backward compatibility only.

SQL Statement Initialization
C/C++
DG_HANDLE InitStatement(DG_HANDLE connection, char *SQL)

C#/VBA/scripts
int InitStatement(int connection, string SQL)
'connection' is valid handle generated by ConnectDB function.
'SQL' is SQL statement definition like 'select AuthorID from Authors order by Name'.

Generate value
C/C++
char* StatementValue(DG_HANDLE handle,int sequentially)

C#/VBA/scripts
string StatementValue(int handle,int sequentially)
Not 0 'sequentially' value means the function will return values from the query results sequentially, random otherwise.

Note: this fill method does not supported by core SDK edition.

Rule level API

Rule level API operates with single rule. In most cases one rule relates to one destination table. Rule is an aggregate of properties and array of fill methods (items) for columns of a table. Each rule item correspond to single column.

- Rule properties operations
- Rule items operations
- One item operations

See also: handles.

GetRuleProperty retrieves property of the specified rule.
C/C++
char* GetRuleProperty(RULE_HANDLE handle,int PropertyID)

C#
string GetRuleProperty(int handle,int PropertyID)
returns property value or empty string for incorrect parameters.

SetRuleProperty set value for the specified property of the rule.
C/C++
int GetRuleProperty(RULE_HANDLE handle,int PropertyID,char *value)

C#
int GetRuleProperty(int handle,int PropertyID,string value)
returns
0 for success
-1 for invalid rule handle
-2 for unknown or unsupported property ID.

COM object offers "RuleProperty" string property for described functions. You should not use "SetRuleProperty" and "GetRuleProperty" functions directly in this case.

Rule Property Definitions

Name	Property ID	Description
RP_tabName	1	Table name to be populated by the rule.
RP_sqlBefore	2	SQL script that will be executed before the rule. ';' or 'go' separator required for complex scripts.
RP_sqlAfter	3	SQL script that will be executed after the rule. ';' or 'go' separator required for complex scripts.
RP_note	4	Rule note (comment)
RP_UpdWhere	5	Where clause for Update mode
RP_sOutFileSQL	6	File name for individual rule SQL output
RP_sOutFileTXT	7	File name for individual rule text output
RP_sTarget	8	Target table name for Scramble Mode
RP_Records	9	Number of rows to be generated by the rule
RP_PerTrans	10	Transaction size or 0 for auto commit
RP_mode	11	Rule type: 0 for Data Rule, 1 for Table Rule, 2 for Object Rule, 3 for File Rule and 4 for Clear Rule
RP_insMode	12	Insert mode: 0 for append, 1 for replace, 2 for Update and 3 for scramble

nRuleItems returns numbers of items (each item corresponds to one field of output data) in the specified rule.
C/C++/C#
int nRuleItems(RULE_HANDLE handle)
Returns number of items (>=0) in success or -1 otherwise.

GetRuleItem returns handle for the specified rule's item.
C/C++/C#
ITEM_HANDLE GetRuleItem(RULE_HANDLE handle, int idx)
'idx' is zero based index of the item to be retrieved.

SetRuleItem replaces existing rule item by specified
C/C++/C#
int SetRuleItem(RULE_HANDLE handle, int idx, ITEM_HANDLE item)
'idx' is zero based index of the item to be replaced.

COM object offers "RuleItem" integer property for described functions. You should not use "SetRuleItem" and "GetRuleItem" functions directly in this case.

RemoveRuleItem removes specified rule item from the list.
C/C++/C#
int RemoveRuleItem(RULE_HANDLE handle, int idx)
'idx' is zero based index of the item to be removed.

AddRuleItem appends new item to the specified rule
C/C++/C#
int AddRuleItem(RULE_HANDLE handle, ITEM_HANDLE item)

SetItemProperty changes selected property of the specified item
C/C++
int SetItemProperty(ITEM_HANDLE handle, int PropertyID, char *value)

C#
int SetItemProperty(int handle, int PropertyID, string value)
returns
0 for success
-1 for invalid rule handle
-2 for unknown or unsupported property ID.

GetItemProperty retrieves property of the specified rule's item.
C/C++
char* GetItemProperty(ITEM_HANDLE handle, int PropertyID)

C#
string GetItemProperty(int handle, int PropertyID)
PropertyID is property code from DG_ITEM_PROPS list (see table below or DGSKD.H).

COM object offers "ItemProperty" string property for described functions. You should not use "SetItemProperty" and "GetItemProperty" functions directly in this case.

Property definitions

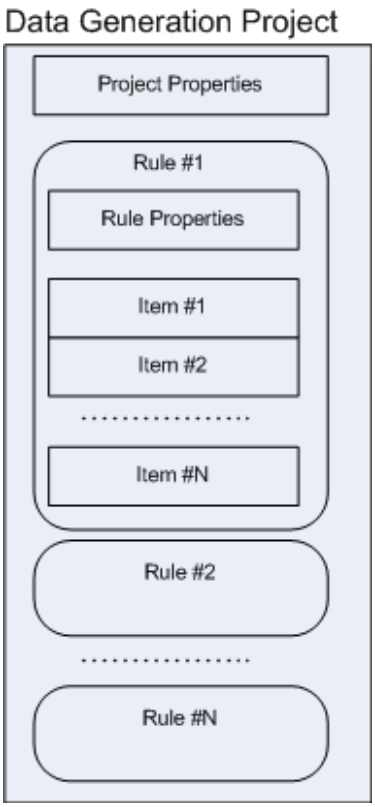
Name	Property ID	Description
IP_Name	1	Column (field) name
IP_Start	2	Start value for incremental fill method
IP_Step	3	Step value for incremental fill method
IP_Times	4	Use each value for incremental fill method
IP_Cycle	5	Cycle length for incremental fill method
IP_AfterLast	6	Insert value after last existing value for incremental fill method
IP_isUnique	8	The library should generate unique values fir this item in case this value is non-zero
IP_isSeqent	9	The library should use values from the list sequentially in case this value is non-zero
IP_method	10	Fill method: 0 to 10
IP_UseNulls	11	% of NULL values
IP_SequenceLen	12	Sequence length. How many times the program should use one generated value
IP_MinLength	13	Minimum value or length
IP_MaxLength	14	Maximum value or length
IP_RndSeparators	16	% separators for random strings
IP_Formula	18	Formula for corresponded fill method
IP_table	19	Linked table for "From Table" fill method
IP_column	20	Linked column for "From Table" fill method
IP_LibName	21	Library item for "From Library" fill method
IP_mask	22	Mask for "By Table" fill method
IP_sFormat	23	Output format string (%d, %5.2f, etc)

IP_Where	24	Where clause for "From Table" fill method
IP_Order	25	"Order by" clause for "From Table" fill method
IP_SQL	26	SQL statement for "By SQL" fill method
IP_incTimeMode	27	Time Mode for incremental method: 0 for second, 1 for minute, 2 for hour, 3 for day, 4 for month or 5 for year

Project level API

Project level API offers a set of functions to operate with a whole project, project's properties and lists of project's rules.

- General operations
- Project properties operations
- Project rules operations



See also: handles.

General project operations

All these functions deal with PRJ_HANDLE that helps you to identify each active project. Current version of the SDK supports up to 8 active projects.

LoadProject function loads project from the disk file.
C/C++
PRJ_HANDLE LoadProject(char *FileName)

C#/VBA/scripts
int LoadProject(string FileName)

SaveProject saves specified by handle project to the disk file.
C/C++
int SaveProject(PRJ_HANDLE handle, char *FileName)

C#/VBA/scripts
int SaveProject(int handle, string FileName)
Returns 0 in success or non-zero otherwise.

NewProject creates new empty project.
C/C++/C#
PRJ_HANDLE NewProject();

CloseProject closes specified project and removes it from RAM.
C/C++/C#
void CloseProject(PRJ_HANDLE handle);

RunProject executes the project with specified database connection.
C/C++
int RunProject(PRJ_HANDLE handle, DG_HANDLE connection, char *logFile)

C#/VBA/scripts
int RunProject(int handle, int connection, string logFile)
RReturns 0 for success or non-zero otherwise.
Notes:

- 'connection' parameter should be correct and connected handle of database connection.
- 'logFile' is location where execution log will be created. The program uses "error.log" in the current directory for empty or invalid parameter.

See also: Error handling.

GetProjectProperty retrieves selected project property.
C/C++
char *GetProjectProperty(PRJ_HANDLE handle,int PropertyID)

C#
string GetProjectProperty(int handle,int PropertyID)

SetProjectProperty sets or changes specified project property.
C/C++
int SetProjectProperty(PRJ_HANDLE handle, int PropertyID, char *value)

C#
int SetProjectProperty(int handle, int PropertyID, string value)
Returns:
0 - executes successfully.
-1 - invalid handle
-2 - passed handle does not allocated.
-4 - unknown property ID.

COM object offers "ProjectProperty" string property for described functions. You should not use "SetProjectProperty" and "GetProjectProperty" functions directly in this case.

Property definitions

Name	Property ID	Description
PR_TruncateSQL	1	The library should truncate output SQL file before each project execution in case this value is non-zero.
PR_TruncateTXT	2	The library should truncate output text file before each project execution in case this value is non-zero.
PR_TruncateXML	3	The library should truncate output XML document before each project execution in case this value is non-zero.
PR_ToFile	4	Create output SQL script when this property is non-zero.
PR_ToTxtFile	5	Create output text file when this property is non-zero.
PR_ToXMLFile	6	Create output XML document when this property is non-zero.
PR_TxtOutColumnNames	7	The library should write column names to text file when this property is non-zero.
PR_AddCommit	8	The library should add COMMIT statements to output SQL script in case this value is non-zero.
PR_NoExec	9	The library will not modify database and create output file(s) only in case this value is non-zero.
PR_QuoteExportString	10	For output text file this property switches quotation node. All strings will be quoted when this property is non-zero.
PR_sqlBefore	11	SQL script that will be executed before project. ';' or 'go' separator required for complex scripts.
PR_sqlAfter	12	SQL script that will be executed after project. ';' or 'go' separator required for complex scripts.
PR_SQLFile	13	File name for SQL output.

PR_TXTFile	14	File name for text output.
PR_XMLFile	15	File name for XML output.
PR_StmtDelimiter	16	SQL statement delimiter for SQL output.
PR_TxtDelimiter	17	Text items delimiter for text output.

nRules returns number of rules in the project or -1 for error.
C/C++/C#
int nRules(PRJ_HANDLE handle)
Returns >=0 in success or -1 otherwise.

GetRule returns handle of the specified rule.
C/C++/C#
RULE_HANDLE GetRule(PRJ_HANDLE handle, int idx)
'idx' is zero based index of the rule to be returned.
Returns handle in success or 0 otherwise.

SetRule replaces existing project's rule by specified.
C/C++/C#
int SetRule(PRJ_HANDLE handle, int idx, RULE_HANDLE rule)
'idx' is zero based index of the rule to be replaced.
Returns
0 for success
-1 for invalid project handle
-2 for not allocated project handle
-4 for invalid rule index 'idx'.

COM object offers "Rule" integer property for described functions. You should not use "SetRule" and "GetRule" functions directly in this case.

AddRule appends new rule to specified project.
C/C++/C#
int AddRule(PRJ_HANDLE handle, RULE_HANDLE rule)

'rule' is a handle of rule that will be added to the project.
Returns
0 for success
-1 for invalid project handle
-2 for not allocated project handle.

RemoveRule removes rule from the project by index.
C/C++/C#
int RemoveRule(PRJ_HANDLE handle,int idx)
'idx' is zero based index of the rule to be removed.
Returns 0 in success, -1 for invalid handle or -2 for invalid rule index.

Object handles

Handle is a session-unique integer identifier or all SDK objects like connections, fill methods, rules, etc. The user should close unnecessary handles by **CloseH** call.

To close project handle the user should call **CloseProject** function and pass handle as parameter.

To close rule the user should call **CloseRule** function and pass rule handle as parameter.

To close rule's item the user should call **CloseItem** function and pass handle as parameter.

Note: a few functions return 0 is invalid handle. Please call GetError function for details in this case.

Handle types

Type	Description / Use by
DG_HANDLE	Fill methods and database connections
PRJ_HANDLE	Project handle
RULE_HANDLE	Rule handle
ITEM_HANDLE	Rule's item handle

Database connections

The standard and professional versions of SDK allows the user to create a few database connections. The connection can be used to generate test data as a source of the data. Please refer to from table and by SQL fill methods for details

Connect to database
C/C++
DG_HANDLE ConnectDB(char *DataSource, char *user, char *Password, char *database)

Important: please use empty strings instead of null pointers for optional parameters.

C#/VBA/scripts
int ConnectDB(string DataSource, string user, string Password, string database)

'DataSource' is name of predefined ODBC data source. Please use ODBC Administrator (odbcad32.exe or Control Panel icon) to prepare this name.
'user' and 'password' are optional and define login (user name) that will be used for connection.
'database' is optional as well and can be use with Microsoft SQL server only.

Execute SQL statement
C/C++
int ExecDB(DG_HANDLE connection,char *sql)

C#

int ExecDB(int connection,string sql)

Note: current SDK edition does not allow the user to operate with result set of this function. You should use it with INSERT, DELETE, UPDATE or procedure call SQL statements only.

Disconnect
C/C++/C#
void DisconnectDB(DG_HANDLE connection)

Error handling

Zero value as a handle means error. The user can call GetError function to retrieve error description.

C/C++
char* GetError(HANDLE handle)

C#/VBA/scripts
string GetError(int handle)

Notes:

1. zero value is acceptable as parameter in case last called function returns 0 as handle.
2. the function accepts DG_HANGLE, PRJ_HANDLE, RULE_HANDLE, etc as a handle parameter.

SDK limitations

The current version of SDK has a few internal limitations. They are:

- Maximum number of active projects is 8
- Maximum number of active handles for fill methods and connections is 256
- Maximum number of active rules is 1024
- Maximum number of active rule items is 10240

Demo version of SDK shows popup window after each 50 generated values.

How to build your program with Data Generation SDK

There are to ways to all data generation functionality to your script or program. The first way is direct import required functions from DLL. This way is suitable for C/C++ and, in some cases, for C#. The second way is to use COM object "DGSDK.DGLib". This way is suitable for scripting languages like VBA, JScript, etc. You should use it with Active Server Page technologies as well.

Visual C++
At first, you should use #include "dgsdkdefs.h" directive in your source file(s).
The SDK contains "dgsdk.lib" import library that compatible with all modern Visual C++ versions starting 6.0. You should specify it as a linker input.
Example: cl mysource.cpp dgsdk.lib

C#
You can use DllImport to access required SDK functions located in "dgsdk.dll".
Example: csc mysource.cs

VBA
You should create "DGSDK.DGLib" object and use methods and properties to access SDK functionality.
Example:

```
Set obj = CreateObject("DGSDK.DGLib")
 Debug.Print obj.DateRandom("DD/MM/YY", "01/01/80", "12/12/97")
```

Python
You should load "DGSDK.DLL" and use methods and properties to access SDK functionality.
Example:

```
from ctypes import *
import ctypes
 dgSDK = cdll.LoadLibrary("dgsdk.dll")
```

Note: SDK contains command files (.cmd) that helps you build and run included examples by one click in case BIN directory of your C++ installation is included into PATH environment variable.

Important: you should place 'dgsdk.dll' and 'common.dll' in the folder where your application is located or add that folder to PATH environment variable.

Demo and Full versions of the SDK

What differences between the demo and full versions of DTM Data Generator SDK?

- Demo version shows the pop-up window after each 50 calls of each data generation functions.

No other demo limitations are present.

End User License Agreement

This License Agreement covers all existing versions of DTM Data Generation SDK: demo and commercial. License Agreement is a legal agreement between the end-user and DTM soft. By installing or otherwise using DTM Data Generation SDK, 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 SDK.

General Information

DTM soft is exclusive owner of all DTM Data Generation SDK copyrights. The product is licensed, not sold. DTM Data Generation SDK is protected by copyright laws and international copyright treaties. Demo version. Anyone may install and use demo version of DTM Data Generation SDK for evaluation and testing purposes. Commercial version. You may install and use one copy of DTM Data Generation SDK on a single computer. The primary user of the computer on which DTM Data Generation SDK is installed may make a second copy for his or her exclusive use on a portable computer. You may not reverse engineer, modify, decompile, or disassemble DTM Data Generation SDK. The Software Product is licensed as a single product. Its component parts may not be separated for use on more than one computer. You may not rent, lease, or lend DTM Data Generation SDK. Also, You may not resell, or otherwise transfer for value, DTM Data Generation SDK. 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 SDK and all of its component parts. You may permanently transfer all of your rights under this license, provided you retain no copies, you transfer all of DTM Data Generation SDK (including all component parts), and the recipient agrees to the terms of this license.

DTM Data Generation SDK IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE DTM Data Generation SDK 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

Professional SDK license allows the licensee to distribute dgsdk.dll and common.dll with SDK based solutions. Standard and Core license permits to use these SDK components in-house only.

Support and Upgrades

During six months after ordering any license, you are entitled to free technical services and support for DTM Data Generation SDK which is provided to you by DTM soft. During this period, e-mail support is unlimited and includes technical and support questions. Also, during mentioned period, you may access to free updates to DTM Data Generation SDK 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 (six months), you may purchase Upgrade and Support subscription. If you ordered a few SDK licenses, you will access to free upgrade and support period and will use subscriptions independently.

Pattern Initialization
C/C++
DG_HANDLE InitPattern(char *Pattern)

C#
int InitPattern(string Pattern)

'Pattern' is definition of the Pattern that will be used to generate value.

Generate value
C/C++
char* PatternValue(DG_HANDLE handle)

C#
string PatternValue(int handle)

End of row mark
C/C++
void PatternEndOfRow(DG_HANDLE handle)

C#
void PatternEndOfRow(int handle)

Please refer to pattern engine manual for details.