

tnsgen Users Guide

**A product of
tns software inc.,
1 Bridgewater Rd.,
Etobicoke, Ontario
M9A 2Z2
Canada**

<http://www.tns-soft.com>

Introduction

It has been estimated that testing accounts for 50% of the total cost of software development. It has also been estimated that 90% of applications are brought into production without being load tested with appropriate data volumes. Adequate testing (and adequate documentation) are often sacrificed at the end of a project to meet project timelines at the expense of application stability and performance.

In cases where the application requires the conversion of legacy data, the testing is often attempted with converted data. This is often a poor choice for testing as the converted data may itself be erroneous, caused by poor data integrity in the legacy source system or errors in the data conversion. More recently, privacy legislation may prohibit the use of legacy data for testing where security controls may not be present to protect the information from unauthorized access.

The obvious solution to these challenges is the use of a simple test data generator that is capable of generating both large and reasonably complex data sets in a simple and timely manner. This was the impetus behind the creation of **tnsgen**, tns software's extensible, script driven test data generator. We at tns software have been using this generator for several years with great success and have decided to make this tool available as freeware for educational and non-commercial purposes. The freeware version is limited to generating no more than 32 fields per output record, while the commercial version supports the generation of up to 255 fields per output record. Please contact us at info@tns-soft.com if you would like to purchase a license for commercial use. License fees begin at \$99.00 (US) for a single-user copy.

Overview

Tnsgen offers a tremendous degree of control in the format of the generated data and allows input data to be derived from input data files in CSV format, enumerated values within a script file or from internal functions that provide random or sequential generation of numbers or dates. **Tnsgen** also support random generation of specialized data types, including Canadian Postal Codes and Canadian Social Insurance Numbers and several types of randomized text, including "greeked" textual output. **Tnsgen** also includes "seed" data for such common attributes as Names, Addresses, Canadian and US Cities, Provinces, States, Area Codes, Stock Symbols and more. Any data set may be easily created and added to allow virtually limitless data generation.

Supplied Data Files

The following data files are supplied with **tnsgen**.

File Name	Description
apt_floor.csv	Apartment Floors
cdn_area_codes.csv	Canadian Telephone Area Code
cdn_cities.csv	Canadian Cities
cdn_provinces.csv	Canadian Provinces
country.csv	Countries
currency.csv	Currencies
first_names.csv	First Names (male and Female)
first_names_female.csv	Female First Names
first_names_male.csv	Male First Names
passwords.csv	Common Passwords
stocks.csv	Stock Symbols
street_names.csv	Street Names
surname.csv	Surnames
us_area_codes.csv	US Area Codes
Us_cities.csv	US Cities
Us_states.csv	US States

Internal Data Types

The following data-types are internally generated within **tnsgen**.

Name	Description
constant-value	Outputs the constant value defined in the controlling script file
db-sequence	Outputs the Oracle specific definition for a database sequence (field-name.nextval)
file-sequence	Outputs the sequence within this output file
greek	Outputs "greeked" text, similar to "Lorem ipsit ...". Very useful when you don't want the textual content to be considered as part of testing.
host-command	Outputs the value of a host command executed in the local environment. While this allows extending operation to include user-defined functions, the rate of data generation is dramatically decreased when this function is employed.
random-date	Outputs a random date. The upper and lower limits of the date may be bounded and the global date mask may define the output format.
random-datetime	Outputs a random date. The upper and lower limits of the date may be bounded and the global datetime mask may define the output format.
random-float	Outputs a random floating-point number. The upper and lower limits of the date may be bounded and a mask may define the output format.
random-integer	Outputs a random integer number. The upper and lower limits of the date may be bounded and a mask may define the output format.
random-list	Outputs one value from a comma-separated list defined

	within the script file. This is useful where only a small number of possible values exist, such as Y or N
random-po	Outputs a random Canadian Postal Code
random-sin	Outputs a random Canadian SIN which passes LUHN validation
random-string	A random string of characters
random-word	A random English word
record-sequence	Outputs the sequence for this record
run-date	Outputs the current date
sequential-number	Outputs a sequential number. The starting number may be defined, otherwise the initial value is 1.

Additional Directives

These are some of the additional directives within **tnsgen** to provide greater control over the content and format of the generated output.

Name	Scope	Description
conversion	Field	Converts the field value to uppercase, lowercase or initcap
datetime-mask	Options	Determines the date format mask for all random-datetime values.
date-mask	Options	Determines the date format mask for all random-date and run-date values.
dup-prev	Field	Duplicates the value of the previous field. Useful where you wish to have a different representation (e.g. uppercase) of the field

file-footer	File	The Footer written once at the end of the file
file-header	File	The Header written once at the beginning of the file
percent-null	Field	The percentage of null values to generate
record-footer	Record	The footer written at the end of each record
record-header	Record	The header written once at the start of each record
skip-format	Field	Skip all pre-defined formatting for this field
skip-prior-null	Field	If the prior field value was null, skip output of this field also. Useful for creating consistent compound fields, such as addresses.
output-format	File	One of XML, CSV, SQL or NONE. The "canned" output format directives allow the generation of data with little formatting directives.

Usage

Tnsngen is a script driven utility which operates in text mode from the console. By default, tnsngen will load and run the script tnsngen.ini if no script file is specified. Otherwise, tnsngen will load and run the script file specified on the command line. The following example illustrates the use of a user-named script file:

```
tnsngen parameter-file=myfile.ini
```

Sample control files are supplied and commented to show how commands and command options may be used to control the content and format of the generated output files.

Example

The following example illustrates the generation of data for the following database table.

```
create table Donor
(
  Donor_Id Number(10,0)          not null
,First_Name          Varchar2(30)          not null
,Last_NameVarchar2(30)          not null
,Last_Name_Upper    Varchar2(30)          not null
,Address_1Varchar2(30)
,Address_2Varchar2(30)
,Address_3Varchar2(30)
,City      Varchar2(30)          not null
,Province  Varchar2(4)          not null
,Postal_Code Varchar2(11)
,SIN       Number(9)
,date_of_birth      date          not null
,year_to_date_Contributions number(10,2)  not null
,comments varchar2(80)
,primary key ( donor_id )
);
```

Sample Control Script

The following script file control defines corresponding output for the above table:

```
; IMPORTANT

; this is the best source for gaining a better understanding of tnsngen and the options available
; please follow this file carefully to understand the capabilities ( and limitations ) of tnsngen.
;
; gendata.ini - parameter-file to control test data / test case generation
; Copyright 2003 tns software inc. All rights Reserved
;
; This file is divided into three sections:
; [options] - where global parameter are defined
; [input-definition] - where input data is defined
; [output-definition] - where output data is defined
;
; Input files must be in conformant CSV format. A number of CSV sample data files
; have been supplied, but any file may be used in either random or sequential mode
;
; options section for global program options and directives
; =====
[options]
; only useful for debugging ( requires source code )

debug          true

;
; if random-seed is set to true, generated output files will vary from each invocation of tnsngen.
;

random-seed    false

;
; NOTE:
; All generated dates, i.e. run-date, random-date and random-datetime, are formatted according to
; global date formatting masks. The field-mask for a date field may be used to modify the format
; using
; standard c sprint formats, but may not be used to redefine the date format.
;
```

```

; date-mask is similar to strftime and supports the following attributes
;
; %d   Day of month
; %D   Zero padded day of month
; %H   uppercase three character month abbreviation
; %h   initcap three character month abbreviation
; %m   Integer month of year
; %M   Zero padded integer month of year
; %Y   4 digit year
; %y   2 digit year ( COBOL backwards compatibility mode )

date-mask          "%Y/%M/%D"

;
; datetime-mask is similar to strftime and supports all
; the attributes of date-mask with the addition of the following attributes
; %I   Hour ( 0 - 23 )
; %i   Hour ( 0 - 12 )
; %j   Minutes ( 0 - 59 )
; %k   Seconds ( 0 - 59 )

datetime-mask     "%Y-%M-%D %I:%j:%k"

;
; user defined value for null ( normally " " )
;

null-value        (null)

;
; The input-definition section for definitions of input data CSV files
; A definition consists of file-name, field-count and field-names ( spaces-separated)
; Where multiple fields are present, the fields following the first field are not randomized
; within
; A record will retain context based upon a reference to the first field
;
=====

[input-definition]
file-name         ./data/cdn_area_codes.csv
field-count       3
field-names       cdn_area_code cdn_area_city cdn_area_prov

file-name         ./data/cdn_cities.csv
field-count       2
field-names       cdn_city cdn_prov_code

file-name         ./data/passwords.csv
field-count       1
field-names       password

file-name         ./data/country.csv
field-count       2
field-names       country_code country_desc

file-name         ./data/currency.csv
field-count       2
field-names       currency_code currency_desc

file-name         ./data/first_names_female.csv

```

```

Field-count      1
field-names      female_name

file-name        ./data/first_names_male.csv
field-count      1
field-names      male_name

file-name        ./data/first_names.csv
field-count      1
field-names      first_name

file-name        ./data/surname.csv
field-count      2
field-names      surname surname_initial

file-name        ./data/cdn_provinces.csv
field-count      2
field-names      prov_code prov_desc

file-name        ./data/us_states.csv
field-count      2
field-names      state_code state_desc

file-name        ./data/stocks.csv
field-count      2
field-names      ticker_symbol company_name

file-name        ./data/street_names.csv
field-count      1
field-names      street_name

file-name        ./data/us_area_codes.csv
field-count      3
field-names      us_area_code us_area_city us_area_state

file-name        ./data/us_cities.csv
field-count      2
field-names      us_city_name us_state

;
; output definition section where output files, records and fields are defined
; =====
;
[output-definition]
file-name        "donor.dat"

; output mode is append or replace
output-mode      replace

; output-format is XML, SQL, CSV, NEW-LINE or "" for fixed output files
output-format    xml

; record-name for output file. Multiple records may be written to the same output file
record-name      donor
; total records to output
record-count     10000

; file-header is written once at the start of the file
file-header      "<!-- start XML generated data for donor-->\n"

```

```

; file footer is written once at the end of the file
file-footer    "<!-- end XML generated data for donor-->\n"

; record-header is written prior to each record
record-header  "<Donor>\n"

; record-footer is written following each output record
record-footer  "</Donor>\n"

; # of times to repeat each output record within the record-header / record-footer boundaries
repeat-record  1

; field-name (required )
field-name     donor_id

; field-length ( required ). Fields will be truncated to not exceed this limit
field-length   10

; field mask (optional). A printf style mask for output formatting which includes support for
tabs ("\t") and newlines ("\n").
; IMPORTANT - you must match the mask to the appropriate data-type - %d for random-integer,
record-sequence, %f for random-float
; and %s for all other output types
field-mask     "%d"

; field-value (required) is one of
;
; "quoted value" - a constant value specified within quotes
; $VALUE         - use a value found in the calling shell environment.
; &VALUE        - use a value found on the calling command-line, such a VALUE="hello"
; field-name    - a field name from one of the files loaded in the input section, such as surname
; random-date   - a random date bounded by low-value and high-value
; random-datetime - a random datetime bounded by low-value and high-value
; random-string - a random string of characters
; greek        - a random string of latin words in sentence form.
; random-word  - a random string of English words in sentence form.
; random-integer - a random integer optionally bounded by low-value and high-value
; random-float - a random floating-point number optionally bounded by low-value and high value
; sequential-number - a sequential number beginning from a defined seed value
; random-list  - a quoted list of comma-separated values from which a single value will be
randomly chosen
; host-command - a user-defined command which when executed by the host will return character
data
; db-sequence  - an Oracle database sequence ( field-name.nextval )
; file-sequence - the output file sequence
; record-sequence - the output record sequence
; run-date    - the current date using the global date mask
; dup-prev   - the same value as the previous field
; random-sin - a random Canadian SIN which passes LUHN validation
; random-po  - a random Canadian Postal Code in the form A9A9A9
field-value   record-sequence

; percent-null (optional) the percentage of output fields which should contain no value
percent-null  0

; conversion (optional)
; one of upper, lower or initcap

; low-value (optional)
; the lower boundary for the field value if supported by the field-value type

```

```

; high-value (optional)
; the upper boundary for the field value if supported by the field-value type

; skip-formatting
; ignores pre-defined output formatting for XML, SQL and CSV.
; Useful when you want to create a single field from multiple input fields - See address_1
; in this file far an example

; random-values      (optional )
; true or false - either assigns the input CSV values sequentially to the output file
; or randomly selects a value. Useful if you want to ensure complete value coverage
; in a small output file
;
; END of Field option comments
; More field definitions follow without comments
;
field-name      first_name
field-length    30
field-mask      "%s"
field-value     first_name
conversion      none
percent-null    0

field-name      last_name
field-length    30
field-mask      "%.5s"
field-value     surname
conversion      none
percent-null    0

field-name      last_name_upper
field-length    30
field-mask      "%s"
field-value     dup-prev
conversion      upper
percent-null    0

field-name      street_number
field-length    10
field-mask      "<address_1>%d "
field-value     random-integer
percent-null    0
low-value       0
high-value      100000
skip-formatting true

field-name      address_1
field-length    30
field-mask      "%s</address_1>\\n"
field-value     street_name
conversion      none
percent-null    0
skip-formatting      true

field-name      address_2
field-length    30
field-mask      "%s"
field-value     ""

```

```

field-name      address_3
field-length    30
field-mask      "%s"
field-value     ""

field-name      city
field-length    30
field-value     cdn_city
random-values   true
conversion      upper
percent-null    0

field-name      province
field-length    4
field-value     cdn_prov_code
random-values   true
conversion      upper
percent-null    0

field-name      postal_code
field-length    11
field-value     random-po
conversion      upper
percent-null    50

field-name      sin
field-length    9
field-value     random-sin

field-name      date_of_birth
field-length    15
; field-mask    "%s"
field-value     random-datetime
percent-null    0
low-value      "01-jan-1950 01:00:00"
high-value     "31-dec-2003 23:59:59"

field-name      year_to_date_contributions
field-length    10
field-mask      "%.2f"
field-value     random-float
percent-null    10
low-value      0.0
high-value     100000

field-name      Comments
field-length    80
field-value     greek
percent-null    80

field-name      test_list
field-length    20
field-value     random-list "A,B,C,D,E"
percent-null    80

file-name       "donor.dat"
output-mode     append
output-format   sql

```

```

record-name      donor
record-count     100
file-header      "///\n// start sql insert statments generated for table donor\n\n"
file-footer      "///\n// end sql insert statments generated for table donor\n\n"

field-name       donor_id
field-length     10
field-value      record-sequence

field-name       first_name
field-length     30
field-value      first_name

field-name       last_name
field-length     30
field-value      surname

field-name       last_name_upper
field-length     30
field-value      dup-prev
conversion       upper

field-name       street_number
field-length     10
field-value      random-integer
skip-formatting  true
field-mask       "%d "
low-value        0
high-value       100000

field-name       address_1
field-length     30
field-value      street_name
field-mask       "%s"
percent-null     0
skip-formatting  true

field-name       address_2
field-length     30
field-mask       "%s"
field-value      ""

field-name       address_3
field-length     30
field-mask       "%s"
field-value      ""

field-name       city
field-length     30
field-value      cdn_city
conversion       upper

field-name       province
field-length     4
field-value      cdn_prov_code
conversion       upper
percent-null     0

field-name       postal_code
field-length     11

```

```

field-value      random-po
conversion       upper
percent-null     50

field-name       sin
field-length     9
field-value      random-sin

field-name       date_of_birth
field-length     15
field-value      random-date
percent-null     0
low-value        "01-jan-1950"
high-value       "31-dec-2003"

field-name       year_to_date_contributions
field-length     10
field-mask       "%.2f"
field-value      random-float
percent-null     10
low-value        0.0
high-value       100000

field-name       Comments
field-length     80
field-value      greek
percent-null     80

file-name        "donor.dat"
output-mode      append
output-format    csv
record-name      donor
record-count     100
file-header      "--\n-- csv data generated for table donor\n--\n"
file-footer      "--\n-- end csv data generated for table donor\n--\n"

field-name       donor_id
field-length     10
field-value      record-sequence

field-name       first_name
field-length     30
field-value      first_name

field-name       last_name
field-length     30
field-value      surname

field-name       last_name_upper
field-length     30
field-value      dup-prev
conversion       upper

field-name       street_number
field-length     10
field-value      random-integer
skip-formatting  true
field-mask       ", \"%d "
low-value        0
high-value       100000

```

```

field-name      address_1
field-length    30
field-value     street_name
field-mask      "%s\"
percent-null    0
skip-formatting true

field-name      address_2
field-length    30
field-mask      "%s"
field-value     ""

field-name      address_3
field-length    30
field-mask      "%s"
field-value     ""

field-name      city
field-length    30
field-value     cdn_city
conversion      upper

field-name      province
field-length    4
field-value     cdn_prov_code
conversion      upper
percent-null    0

field-name      postal_code
field-length    11
field-value     random-po
conversion      upper
percent-null    50

field-name      sin
field-length    9
field-value     random-sin

field-name      date_of_birth
field-length    15
field-value     random-date
percent-null    0
low-value      "01-jan-1950"
high-value     "31-dec-2003"

field-name      year_to_date_contributions
field-length    10
field-mask      "%.2f"
field-value     random-float
percent-null    10
low-value      0.0
high-value     100000

field-name      Comments
field-length    80
field-value     greek
percent-null    80

```

Sample Execution

The following is a Solaris version demonstration of generation, captured using script.

```
tnsngen: Non-Commercial and Educational Release 1.02 for Solaris 2.6 - 2.8

Copyright (c) tns Software Inc., 1994-2003. All Rights Reserved

visit http://www.tns-soft.com for updates or commercial version

tnsngen-03-Oct-03 17:05:38-I: processing command-line parameters ...
Input File Definitions
File Name Fields Field(s)
=====
./data/cdn_area_codes.csv 3 cdn_area_code,cdn_area_city,cdn_area_prov,
./data/cdn_cities.csv 2 cdn_city,cdn_prov_code,
./data/passwords.csv 1 password,
./data/country.csv 2 country_code,country_desc,
./data/currency.csv 2 currency_code,currency_desc,
./data/first_names_female.csv 1 female_name,
./data/first_names_male.csv 1 male_name,
./data/first_names.csv 1 first_name,
./data/surname.csv 2 surname,surname_initial,
./data/cdn_provinces.csv 2 prov_code,prov_desc,
./data/us_states.csv 2 state_code,state_desc,
./data/stocks.csv 2 ticker_symbol,company_name,
./data/street_names.csv 1 street_name,
./data/us_area_codes.csv 3 us_area_code,us_area_city,us_area_state,
./data/us_cities.csv 2 us_city_name,us_state,

tnsngen-03-Oct-03 17:05:38-I: Pass 1 ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for cdn_area_code from ./data/cdn_area_codes.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for cdn_city from ./data/cdn_cities.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for password from ./data/passwords.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for country_code from ./data/country.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for currency_code from ./data/currency.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for female_name from ./data/first_names_female.csv ...
tnsngen-03-Oct-03 17:05:38-I: Checking data for male_name from ./data/first_names_male.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for first_name from ./data/first_names.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for surname from ./data/surname.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for prov_code from ./data/cdn_provinces.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for state_code from ./data/us_states.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for ticker_symbol from ./data/stocks.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for street_name from ./data/street_names.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for us_area_code from ./data/us_area_codes.csv ...
tnsngen-03-Oct-03 17:05:39-I: Checking data for us_city_name from ./data/us_cities.csv ...

Data File Definitions
File Name Field Name Fields Records
=====
./data/cdn_area_codes.csv cdn_area_code 3 15
./data/cdn_cities.csv cdn_city 2 862
./data/passwords.csv password 1 431
./data/country.csv country_code 2 233
./data/currency.csv currency_code 2 179
./data/first_names_female.csv female_name 1 854
./data/first_names_male.csv male_name 1 884
./data/first_names.csv first_name 1 517
```

```
./data/surname.csv surname 2 1794
./data/cdn_provinces.csv prov_code 2 13
./data/us_states.csv state_code 2 57
./data/stocks.csv ticker_symbol 2 671
./data/street_names.csv street_name 1 7449
./data/us_area_codes.csv us_area_code 3 111
./data/us_cities.csv us_city_name 2 349

tnsngen-03-Oct-03 17:05:39-I: Pass 2 ...
tnsngen-03-Oct-03 17:05:39-I: Loaded 15 records for cdn_area_code from ./data/cdn_area_codes.csv
tnsngen-03-Oct-03 17:05:39-I: Loaded 862 records for cdn_city from ./data/cdn_cities.csv
tnsngen-03-Oct-03 17:05:39-I: Loaded 431 records for password from ./data/passwords.csv
tnsngen-03-Oct-03 17:05:39-I: Loaded 233 records for country_code from ./data/country.csv
tnsngen-03-Oct-03 17:05:39-I: Loaded 179 records for currency_code from ./data/currency.csv
tnsngen-03-Oct-03 17:05:39-I: Loaded 854 records for female_name from ./data/first_names_female.cs
tnsngen-03-Oct-03 17:05:39-I: Loaded 884 records for male_name from ./data/first_names_male.cs
tnsngen-03-Oct-03 17:05:40-I: Loaded 517 records for first_name from ./data/first_names.cs
tnsngen-03-Oct-03 17:05:40-I: Loaded 1794 records for surname from ./data/surname.cs
tnsngen-03-Oct-03 17:05:40-I: Loaded 13 records for prov_code from ./data/cdn_provinces.csv
tnsngen-03-Oct-03 17:05:40-I: Loaded 57 records for state_code from ./data/us_stats.csv
tnsngen-03-Oct-03 17:05:40-I: Loaded 671 records for ticker_symbol from ./data/stocks.csv
tnsngen-03-Oct-03 17:05:40-I: Loaded 7449 records for street_name from ./data/street_names.csv
tnsngen-03-Oct-03 17:05:40-I: Loaded 111 records for us_area_code from ./data/us_area_codes.csv
tnsngen-03-Oct-03 17:05:40-I: Loaded 349 records for us_city_name from ./data/us_cities.csv
tnsngen-03-Oct-03 17:05:40-I: All datafiles loaded
tnsngen-03-Oct-03 17:05:40-I: Writing OutputFile <donor.dat> ...
.....
tnsngen-03-Oct-03 17:05:45-I: Wrote 10000 records.
tnsngen-03-Oct-03 17:05:45-I: Writing OutputFile <donor.dat> ...
tnsngen-03-Oct-03 17:05:45-I: Wrote 100 records.
tnsngen-03-Oct-03 17:05:45-I: Writing OutputFile <donor.dat> ...
tnsngen-03-Oct-03 17:05:45-I: Wrote 100 records.
```

Sample Output File

The following is a sample output file in XML format.

```
<!-- start generated data for donor-->
<Donor>
<donor_id>1</donor_id>
<first_name>Kurt</first_name>
<last_name>Nadaue</last_name>
<last_name_upper>NADAUE</last_name_upper>
<address_1>6838 E 42ND ST</address_1>
<address_2></address_2>
<address_3></address_3>
<city>COLCHESTER</city>
<province>ON</province>
<postal_code>(null)</postal_code>
<sin>883517096</sin>
<date_of_birth>03-Nov-1987</date_of_birth>
<year_to_date_contributions>16838.58</year_to_date_contributions>
<Comments>Laoreet eum molestie dignissim te tincidunt. Adipiscing commodo dignissim
acc</Comments>
</Donor>
<!-- end generated data for donor-->
```

Log File

A log file is generated for each execution of the generator. The log file name is always `tnsngen_log.xml` and is located in the working directory. Output in the log file includes additional information and statistics about the values written. The following is a sample log file:

```
<tnsngenOutput>
  <tnsngenVersion>1.01</tnsngenVersion>
  <tnsngenPlatform>Solaris 2.8</tnsngenPlatform>
  <DateRun>2003-08-12 19:52:34</DateRun>
  <OutputRecord>
    <FileName>donor.dat</FileName>
    <RecordName>donor</RecordName>
    <RecordCount>10000</RecordCount>
    <FieldRecord>
      <FieldName>donor_id</FieldName>
      <FieldType>record-sequence</FieldType>
    </FieldRecord>
    <FieldRecord>
      <FieldName>first_name</FieldName>
      <FieldType>external</FieldType>
      <ExternalValue>first_name</ExternalValue>
      <AverageLength>6</AverageLength>
      <MaximumLength>30</MaximumLength>
    </FieldRecord>
    <FieldRecord>
      <FieldName>last_name</FieldName>
      <FieldType>external</FieldType>
      <ExternalValue>surname</ExternalValue>
      <AverageLength>6</AverageLength>
      <MaximumLength>30</MaximumLength>
    </FieldRecord>
    <FieldRecord>
      <FieldName>last_name_upper</FieldName>
      <FieldType>dup-prev</FieldType>
      <FieldConversion>upper</FieldConversion>
    </FieldRecord>
    <FieldRecord>
      <FieldName>street_number</FieldName>
      <FieldType>random-integer</FieldType>
      <LowValue>0</LowValue>
      <HighValue>100000</HighValue>
      <MinimumValue>42</MinimumValue>
      <MaximumValue>98289</MaximumValue>
      <AverageValue>47974</AverageValue>
    </FieldRecord>
    <FieldRecord>
      <FieldName>address_1</FieldName>
      <FieldType>external</FieldType>
      <ExternalValue>street_name</ExternalValue>
      <AverageLength>10</AverageLength>
      <MaximumLength>30</MaximumLength>
    </FieldRecord>
    <FieldRecord>
      <FieldName>address_2</FieldName>
```

```

    <FieldType>constant</FieldType>
    <ConstantValue></ConstantValue>
</FieldRecord>
<FieldRecord>
  <FieldName>address_3</FieldName>
  <FieldType>constant</FieldType>
  <ConstantValue></ConstantValue>
</FieldRecord>
<FieldRecord>
  <FieldName>city</FieldName>
  <FieldType>external</FieldType>
  <ExternalValue>cdn_city</ExternalValue>
  <AverageLength>10</AverageLength>
  <MaximumLength>30</MaximumLength>
  <FieldConversion>upper</FieldConversion>
</FieldRecord>
<FieldRecord>
  <FieldName>province</FieldName>
  <FieldType>external</FieldType>
  <ExternalValue>cdn_prov_code</ExternalValue>
  <AverageLength>2</AverageLength>
  <MaximumLength>4</MaximumLength>
  <FieldConversion>upper</FieldConversion>
</FieldRecord>
<FieldRecord>
  <FieldName>postal_code</FieldName>
  <FieldType>random-po</FieldType>
  <FieldConversion>upper</FieldConversion>
  <NullCount>5561</NullCount>
  <NullTargetPct>50</NullTargetPct>
  <ActualNullPct>55</ActualNullPct>
</FieldRecord>
<FieldRecord>
  <FieldName>sin</FieldName>
  <FieldType>random-sin</FieldType>
</FieldRecord>
<FieldRecord>
  <FieldName>date_of_birth</FieldName>
  <FieldType>random-date</FieldType>
  <LowValue>01-jan-1950</LowValue>
  <HighValue>31-dec-2003</HighValue>
</FieldRecord>
<FieldRecord>
  <FieldName>year_to_date_contributions</FieldName>
  <FieldType>random-float</FieldType>
  <LowValue>0.00</LowValue>
  <HighValue>100000.00</HighValue>
  <MinimumValue>3.50</MinimumValue>
  <MaximumValue>98262.78</MaximumValue>
  <AverageValue>40094.30</AverageValue>
  <NullCount>1839</NullCount>
  <NullTargetPct>10</NullTargetPct>
  <ActualNullPct>18</ActualNullPct>
</FieldRecord>
<FieldRecord>
  <FieldName>Comments</FieldName>
  <FieldType>greek</FieldType>
  <AverageLength>12</AverageLength>
  <MaximumLength>80</MaximumLength>
  <NullCount>8508</NullCount>

```

```
<NullTargetPct>80</NullTargetPct>  
<ActualNullPct>85</ActualNullPct>  
</FieldRecord>  
</OutputRecord>  
</tnsngenOutput>
```

Platforms

Tnsngen is available in binary form for Solaris 2.6-2.8, HP/UX 11i, Red Hat 7.3 and Windows 98/2000 using Cygwin (available from <http://www.cygwin.com>). **Tnsngen** can generate thousands of records per minute even on relatively low-powered workstations.

Availability

Tnsngen is available for download in binary form from <http://www.tns-soft.com>.