CONTENTS

OVERVIEW

WHAT IS THE GENERATOR?

The Generator is a data generation tool designed to make the creation of test or production data for any type of application easy and quick.

HOW DOES THE GENERATOR WORK?

The main component of The Generator is the engine compiler/engine that generates output files from a user-defined specification, a script that the engine reads to generate output. The spec can be created with The Generator, any type of editor, word processor, or scripting tool.

WHAT TYPE OF DATA CAN THE GENERATOR CREATE ?

The Generator has internal algorithms for generating values for dates, times, and numeric values, internal data tables that generate values for some of the more common values such as addresses, cities, states, zip codes, phone numbers, salutations, prefixes, postfixes, etc. , and any type of user-supplied data.

THE GENERATOR ENGINE

The Generator engine is contained in the executable file DBG0.EXE. (That is the number ZERO after G).
The engine looks for a default file name of GENSPEC.TXT or any file name passed as a command line argument.
For example - DBG0 SAMPLE.TXT

In most cases, you will not run the engine from the command line. The GENERATE option of the Spec Form MENU will do this automatically.

THE FILE SPECIFICATION

A specification is a text file that is read by The Generator at run-time. The sample specification below is for a file with 1000 records containing 4 fields, last names, first names, middle names, and a prefix.

```
* This is a comment line
NAMES
1000
F
S
L
1,Last_N,CHAR,NAMEL,   15,R,0,0,0,0,1,4975
2,First_N,CHAR,NAMEF,   15,R,0,0,0,0,1,2876
3,Middle_N,CHAR,NAMEM, 15,R,0,0,0,0,1,5216
4,Prefix,CHAR,PREFX, 2, R,0,0,0,0,1,1366
```

Again, this file can be created with the interface provided with The Generator. You can also create these specifications from scratch in an editor.

THE PARTS OF A SPECIFICATION

A specification contains two parts.

The first part of a specification is the comments area where comments can be entered about the specification. Each comment line must be preceded by an asterisk (*). Comments can appear on any line in a specification and can be used to document the specification. The first part of a specification also contains the File Structure information in 5 lines.

Line 1 is the file name (8 characters, no extension) used by The Generator to create an output file name.
Line 2 is the number of records to be generated.
Line 3 is the file/record type (fixed or variable).
Line 4 is the field delimiter.
Line 5 is the record delimiter.

The second part of a specification contains a line for each field to be generated. Each line contains 12 items, each separated by a comma.

```
* Part 1
* This is a comment line
 NAMES
1000
F
S
L
* Part 2
1,Last_N,CHAR,NAMEL,   15,R,0,0,0,0,1,4975
2,First_N,CHAR,NAMEF,   15,R,0,0,0,0,1,2876
3,Middle_N,CHAR,NAMEM, 15,R,0,0,0,0,1,5216
4,Prefix,CHAR,PREFX, 2, R,0,0,0,0,1,1366
```

DATA SOURCES

The Generator creates output data from three sources.

Supplied Data comes in the form of five encrypted files which contain names, addresses, cities, states, zip codes, phone numbers, gender, and company names.

Internally Generated Data originates from internal algorithms within the engine and includes such things as dates, times, and integers.

User Defined Data can be generated from input files created by the user. The Generator is supplied the file name in the spec file, and at run-time, the file is opened and the values are read into the target field. This gives the user unlimited options for creating any type of input data.

THE GRAPHICAL USER INTERFACE (GUI)

The Generator for Windows comes with a graphical user interface that guides the user through the build process for a specification file. The user can create a specification file using the GUI or build a spec file using any text editor. The GUI provides the field checks and editing validation that an editor does not.
The GUI presents the user with a File Spec form and a Field Spec form. These screens enable the user to quickly build a specification file. If certain types of database files already exist, the user can select the file, and the file structure information will be read into the file spec automatically. The user only needs to define the data-fill types for each field.

THE MAIN MENU

The Main Menu has three selections.

FILE - allows the user to create a new spec, open an existing spec, or create a spec from an existing database.

Under FILE there are three options.

NEW SPEC - takes the user to a new file spec form

OPEN SPEC - opens an existing file spec form

READ DB HEADER - reads database header information and creates a new file spec.

VIEW - is a text file viewer for looking at the output file created by The Generator.

HELP - provides the user on-line help for The Generator.

THE NEW SPEC OPTION

The New Spec Option takes the user to the Spec Form. The user can enter information about the file spec, including Comments,   Spec Name,   # Records,   Record Type,   Field Delimiter,   Record Delimiter,   and each field.

THE OPEN SPEC OPTION

The Open Spec Option lets the user browse the directory and select a Spec that was previously created.   All spec files have an extension of   .SPF.

THE READ DB HEADER OPTION

The Read DB Header Option allows the user to select from 6 database types.   If a database file of that type is selected,   the header information is converted to a partial spec file automatically.   The user must still edit each field and select the data-fill types, bounds, intervals, etc.

SELECT DATABASE FILE TO READ

Select Database File To Read allows the user to select a database type from 6 options and the file. The Generator will read in Field Number, Field Name, and Field Type, converting these values into a partial spec file. The user must complete each field spec in order to generate a spec file.

THE FILE SPEC

The File Spec form is where the user enter global information about the spec file. This includes a comments section, an output file name, the number of records to be generated, the file type, and record and field delimiters.

COMMENTS

The Comments section provides a place for the user to document the specification.   This might include the date, application name,   reason for generation, etc.

SPEC NAME

SPEC NAME is used to create an output file name.
A valid name is a maximum of 8 characters in length. An extension of .GEN is automatically appended at generation time to the file name.

NUMBER (#) RECORDS

# RECORDS is the number of records to be generated. A valid range is 1 to 999,999,999.

RECORD TYPE

RECORD TYPE is the file/record type (fixed or variable). The valid types are Fixed-length or Variable-length records.

FIELD DELIMITER

FIELD DELIMITER is the delimiter placed between fields.   The valid types are (A)postrophe, (C)omma, (S)lash, (P)ipe, (N)ull, and Carriage Return -
(L)ine Feed.

RECORD DELIMITER

RECORD DELIMITER is the delimiter placed between records. The valid types are (A)postrophe, (C)omma, (S)lash, (P)ipe, (N)ull, and Carriage Return - (L)ine Feed.

THE FIELD SPEC

The Field Spec is where the user enters information about each field or where information is read in from a database file header. The Field Spec includes entries for Field Name, Field Type, Data-Fill Type,
Field Width, Order Type, Lower Bound, Upper Bound, Interval Count, Decimal Places, Repeat Value, and Seed Value, Input File Name or String.

FIELD SPEC VALUES

Field 1 is the field number.   A valid number is 1 - 255.
Field 2 is the field name read in from the DB header and is optional.
Field 3 is the field type converted to a Microsoft Visual Basic data type and is optional.
Field 4 is the data-fill type. A list of valid types is listed below.
Field 5 is the field length. The valid lengths are 1 - 255.
Field 6 is the order type. The valid types are (R)andom, (S)equential, and (U)nique.
Field 7 is the lower bounds and valid for numeric and date data-fill types.
Field 8 is the upper bounds and valid for numeric and data fill-types.
Field 9 is the interval count between numeric and date sequential values.
Field 10 is the number of decimal places and the valid range depends upon platform.
Field 11 is the repeat value for numeric and date fields and can be repeated in the next record.
Field 12 is a field for seed values, or an input file name, or a string.

When building a spec file with an editor,   each field must be separated by a comma.

DATA-FILL TYPES

The Generator can create all types of data and the opportunities to be creative are unlimited. The Generator accomplishes this by organizing data into Data-Fill types.

The three general categories of Data-Fill Types are

1) Supplied Data

2) Internally Generated Data

3) User Defined Data

SUPPLIED DATA

The Supplied Data files are encrypted to insure the integrity of the data. The files contain -

Company - company names
Address - street address
City,State,Zip - city, state, zip, area code, and exchange
Names - first, prefix (Mr, Ms), last names, middle names and gender.

INTERNALLY GENERATED DATA

Internally generated data is a value created from algorithms such as integers, dates, and times.

USER-DEFINED DATA

The user can use three different options with user defined data files.

CUSTA should be user created input files that contain alpha-numeric characters and will be left-justified.

CUSTN should be input files with numeric values and are right-justified in the output file.

READF will read into a multi-field record from an offset and for a given length. For example, to read an employee file that has eight fields, and you only want to read field seven, you must supply the offset and field length of field seven.

LIST OF DATA-FILL TYPES

Data-Fill Types comprise the core of The Generator. Traditional file structures use data types such as character, integer, date, etc. In The Generator, the data-fill type is used to determine the output. This saves the user hours and hours of assembling many of the data values most commonly used in a database or program.

ADDR1	Address Line
BLANK	Blank field
CITUS	US cities
COMPA	Company Names
COPYF	Copy a previous field
DAMDY	MM/DD/YY
DAYMD	YY/MM/DD
DMDY4	DDMMYYYY
DY4MD	YYYYMMDD
GENDR	Gender
INTER	Integers
NAMEF	First Names
NAMEL	Last Names
NAMEM	Middle Names
NAMMI	Middle Initials
PHOAC	3 digit area code
PHOEX	3 digit exchange
PHONE	10 digit phone number
PHONO	4 digit phone suffix
PREFX	Name Prefix
STATA	State Abbreviations
STRIN	Any string/constant
T24HR	HHMMSS
ZIP05	Zip Codes

USER-DEFINED DATA-FILL TYPES

THE SPECIAL DATA-FILL TYPES: CUSTA, CUSTN, READF

The special data-fill types are:

CUSTA - custom alpha-numeric input file. When this data-fill type is used, an input file containing a list of data values is read one record at a time, either sequentially or randomly. The values are left-justified.

CUSTN - custom numeric input file. When this data-fill type is used, an input file containing a list of data values is read one record at a time, either sequentially or randomly. The values are right-justified.

READF - read data values in from a file containing records with multiple fields. The user provides the file name, offset and field length in field 12.

DATA-FILL DESCRIPTIONS

A description of each data-fill type follows including a sample of the output generated by that type. Some data-fill types contain a data link to another data-fill type. This insures the data integrity between logical fields such as cities, states, and zip codes, or first names and gender and prefix.

ADDR1

Default Length: 30
Description: Street Address Line
Data Link: None
Sample Output:

```
860 S Los Angeles St 8th Fl
75 Wall St 22nd Fl
750 Central Expy
3611 14th Ave
703 Main St
2707 S East Ave
307 North St
79 5th Ave
3100 Alfred St
190 Independence Dr
```

BLANK
Default Length: 1
Description: Creates an empty/blank field
Data Link: None
Sample Output: None

CITUS

Default Length: 30
Description: US Cities
Data Link: State, Zip, Area Codes
Sample Output:

```
Saint Petersburg,FL,33701
Rock Hill,SC,29730
Myrtle Beach,SC,29577
New Hyde Park,NY,11040
Maspeth,NY,11378
San Francisco,CA,94117
Albuquerque,NM,87108
Richmond,VA,23219
Rochester,MN,55902
Detroit,MI,48211
```

COMPA

Default Length: 30
Description: Company Names
Data Link: None
Sample Output:

```
TSC Div Harper Lloyd Inc
Sofinnova Inc
Software Systems Technology In
Manhattanville College
Sequoia Pacific Systems Corp
Ingram Micro D Inc
Kindel & Anderson
Standard Cabinet Works Inc
Klein Niel K Advertising
Elle Magazine
```

COPYF

Default Length: 1
Description: Copies the value of a previous field
Data Link: None
Sample Output: (Place the number of the field to be copied in field spec, field 12)

CUSTA

Default Length: 1
Description: User created text file
Data Link: None
Sample Output: (Input file contained random alpha characters)

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
IJKLMNOPQRSTUVWXYZABCDEFGH
JKLMNOPQRSTUVWXYZABCDEFGHI
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
YZABCDEFGHIJKLMNOPQRSTUVWX
HIJKLMNOPQRSTUVWXYZABCDEFG
NOPQRSTUVWXYZABCDEFGHIJKLM
KLMNOPQRSTUVWXYZABCDEFGHIJ
EFGHIJKLMNOPQRSTUVWXYZABCD
```

CUSTN

Default Length: 1
Description: User created numeric file
Data Link: None
Sample Output: (Input file contained numbers, right justified in field)

```
001002003
001002002
001002001
001002003
001002004
001002009
001002002
001002005
001002001
001002004
```

DAMDY

Default Length: 8
Description: MM/DD/YY
Data Link: None
Sample Output:

```
09/18/64
08/09/41
09/11/36
02/10/21
03/04/63
02/05/49
01/19/22
05/05/46
05/07/24
08/21/56
```

DAYMD

Default Length: 8
Description: YY/MM/DD
Data Link: None
Sample Output:

```
48/10/25
30/09/14
27/10/15
16/02/14
48/03/05
37/02/07
16/01/27
34/06/08
18/05/10
42/08/30
```

DMDY4

Default Length: 8
Description: MMDDYYYY
Data Link: None
Sample Output:

```
10251759
09141125
10150999
02140601
03051746
02071363
01270615
06081263
05100659
08301538
```

DY4MD

Default Length: 8
Description: YYYYMMDD
Data Link: None
Sample Output:

```
17591025
11250914
09991015
06010214
17460305
13630207
06150127
12630608
06590510
15380830
```

GENDR

Default Length: 1
Description: Output gender in form M or F
Data Link: First Name must also be used to get accurate output
Sample Output:

```
Doyle,M
Theresa,F
Melissa,F
Dargie,M
Renny,M
Sylvia,F
Lorretta,F
Julius,M
Micki,F
Melva,F
```

INTER (POSITIVE)

Default Length: 9
Description: Outputs positive integers
Data Link: None
Sample Output: (6 digit numbers)

```
308655
364996
898512
112114
902794
284925
491744
393979
155818
394186
```

INTER (NEGATIVE)

Default Length: 9
Description: Outputs negative integers
Data Link: None
Sample Output: (6 digit numbes)

```
-69134
-63500
-91014
-88788
-97206
-71507
-50825
-60602
-84418
-60581
```

NAMEF

Default Length: 15
Description: First Names
Data Link: None
Sample Output:

```
Richard
Oran
Orin
Fritz
Neil
Conrad
Doran
Phil
Doyle
Theresa
```

NAMEL

Default Length: 15
Description: Last Names
Data Link: None

Sample Output:

```
Swidler
Slayden
Slocum
Marino
Sergeant
Kerekes
Lang
Stacker
Larned
Gai
```

NAMEM

Default Length: 15
Description: Middle Names
Data Link: None
Sample Output:

```
Richard
Oran
Orin
Fritz
Neil
Conrad
Doran
Phil
Doyle
Theresa
```

NAMMI

Default Length: 1
Description: Middle Initial
Data Link: None
Sample Output:

```
W
V
V
O
V
L
M
W
N
H
```

PHOAC

Default Length: 3
Description: Area Code
Data Link: State, Exchange, City, Zip
Sample Output: (Shown with City, State, Area Code)

```
 Saint Petersburg,FL,813
Rock Hill,SC,803
Myrtle Beach,SC,803
New Hyde Park,NY,516
Maspeth,NY,718
San Francisco,CA,415
Albuquerque,NM,505
Richmond,VA,804
Rochester,MN,507
Detroit,MI,313
```

PHOEX

Default Length: 3
Description: Exchange
Data Link: State, Exchange, City, Zip
Sample Output: (Shown with City, State, Complete Phone Number)

```
Saint Petersburg,FL,8138938803
Rock Hill,SC,8033668344
Myrtle Beach,SC,8034498352
New Hyde Park,NY,5164885629
Maspeth,NY,7183668109
San Francisco,CA,4156684521
Albuquerque,NM,5052654997
Richmond,VA,8046488498
Rochester,MN,5072885039
```

PHONE

Default Length: 10
Description: Area Code, Exchange, Phone Number
Data Link: State, Exchange, City, Zip
Sample Output: Shown with City, State, Complete Phone Number

```
Saint Petersburg,FL,8138938803
Rock Hill,SC,8033668344
Myrtle Beach,SC,8034498352
New Hyde Park,NY,5164885629
Maspeth,NY,7183668109
San Francisco,CA,4156684521
Albuquerque,NM,5052654997
Richmond,VA,8046488498
Rochester,MN,5072885039
Detroit,MI,3138733007
```

PHONO

Default Length: 7
Description: Exchange, Phone Number
Data Link: State, Exchange, City, Zip
Sample Output: Shown with City, State, Complete Phone Number

```
Saint Petersburg,FL,8138938803
Rock Hill,SC,8033668344
Myrtle Beach,SC,8034498352
New Hyde Park,NY,5164885629
Maspeth,NY,7183668109
San Francisco,CA,4156684521
Albuquerque,NM,5052654997
Richmond,VA,8046488498
Rochester,MN,5072885039
Detroit,MI,3138733007
```

PREFX

Default Length: 2
Description: Mr or Ms
Data Link: First Name, Gender
Sample Output:

```
Doyle,MR
Theresa,MS
Melissa,MS
Dargie,MR
Renny,MR
Sylvia,MS
Lorretta,MS
Julius,MR
Micki,MS
Melva,MS
```

READF

Default Length: 1
Description: Copies a string of characters from a file and record based on offset and length.
Data Link: None
Sample Output: (Enter   the file name, offset into the record,   and length of string to be copied)

STATA

Default Length: 2
Description: US State Abbreviations
Data Link: State, City, Zip, Phone Numbers
Sample Output:

```
Saint Petersburg,FL,33701
Rock Hill,SC,29730
Myrtle Beach,SC,29577
New Hyde Park,NY,11040
Maspeth,NY,11378
San Francisco,CA,94117
Albuquerque,NM,87108
Richmond,VA,23219
Rochester,MN,55902
Detroit,MI,48211
```

STRIN

Default Length: 1
Description: Any alpha-numeric string of characters
Data Link: None
Sample Output: (Where the field constant is SAMPLE)

```
Riehard,Swidler,SAMPLE
Oran,Slayden,SAMPLE
Orin,Slocum,SAMPLE
Fritz,Marino,SAMPLE
Neil,Sergeant,SAMPLE
Conrad,Kerekes,SAMPLE
Doran,Lang,SAMPLE
Phil,Stacker,SAMPLE
Doyle,Larned,SAMPLE
Theresa,Gai,SAMPLE
```

T24HR

Default Length: 6
Description: 24 hour time in format HHMMSS or HHMM or HH
Data Link: None
Sample Output:

```
215050
134827
115030
071328
201711
161313
070653
153216
072920
184458
```

ZIP05

Default Length: 5
Description: 5 digit zip code
Data Link: City, State, Zip, Phone Number
Sample Output:

```
Saint Petersburg,FL,33701
Rock Hill,SC,29730
Myrtle Beach,SC,29577
New Hyde Park,NY,11040
Maspeth,NY,11378
San Francisco,CA,94117
Albuquerque,NM,87108
Richmond,VA,23219
Rochester,MN,55902
Detroit,MI,48211
```

DATA-FILL RANGES

Listed below are the default, maximum, and minimum values. These limitations do not apply to the versions of The Generator for MVS, VM, AS/400, or UNIX.

Maximum number of custom files - 255
Maximum length of any field - 255
Maximum printable hours - 23
Maximum printable minutes - 59
Maximum printable seconds - 59
Maximum number of output fields - 512
Maximum years 2 digit - 99
Maximum years 4 digit - 9999
Maximum months - 12
Minimum date 6 digit - 010100
Minimum date 8 digit - 01010008
Maximum date 6 digit - 123199
Maximum date 8 digit - 12319999
Maximum positive integer - 999,999,999
Minimum negative integer - -999,999,999
Maximum limit 4 digit phone number - 9999
Abbreviations for PREFX - Mr, Ms
Abbreviations for GENDR - M,F
Interval Minimum - 1
Interval Maximum - 999,999
Maximum decimal places - 6

Note:
Sequence, Lower, Upper and Interval only apply to numeric data-fill types. The seed value only applies when used with random (R) sequences. Intervals are a minimum of 1.

HOW TO - TIPS & TECHNIQUES

The File Spec can be used to generate complete files containing many fields, or it can be used to build a single complicated field made up of alpha-numeric characters.

This allows for greater flexibility of use and creativity when it comes to producing the type of output desired. To standardize output think about the following guidelines.

The most universally accepted file type is a fixed length record. In most cases the field delimiter can be set to NULL and the record delimiter set to Carriage Return-Line Feed.

Use a specification to develop a single field if the field is a combination of other data types or data-fill types.

Generate a small subset of data and view it before generating large sets of data.

Save the specification, not the output. Once you have finished using the file, delete it. This will save disk space.

Develop sample specifications for less sophisticated users and let them edit the specifications.

Develop scripts in other languages to automatically generate specifications and run The Generator.

HOW TO COMBINE DATA-FILL TYPES

Very often a special combination of values needs to be generated. There are two ways to approach these requirements.

The first option is to extract data from another source and use the CUSTA or CUSTN fill types and reference the file name in field 12.

The second option is to use The Generator to create the field a character or string at a time using the same approach as if one were building multiple fields.

For example, if a field needed to include first name , a comma and a middle initial proceed as follows:

1 - Build a spec using NAMEF, STRING (inserting the comma), and NAMMI.

2 - Generate this output file with the appropriate field and record delimiters.

3 - Use this generated file as a CUSTA data-fill type and include it in the original spec.

In another case, an alpha-numeric sequence of numbers may need to be generated. Again, build a spec using INTER and NAMMI a character at a time to generate the combination of numbers and characters. In fact, STRING and CUSTA and CUSTN could also be used to limit the range of custom values to a small subset. So, not only can a spec be used to build a sequence of fields, but a spec can be used to create a special field type based on the user requirements.

HOW TO GENERATE FREQUENCY DISTRIBUTIONS

Many times there is a requirement to create a sampling of data that may require a frequency of distribution. The Generator can create frequency distributions through two variables.

The first variable is the number of records to be generated. The larger this number, the better the distribution in the output file.

The second variable is the input file or data range. For example, if a field is to contain titles of a company, and 80% of the company is comprised of ANALYSTS, 18% are MANAGERS and 2% are DIRECTORS, then you would want to create a CUSTA input file that has a list with ANALYSTS repeated 80 times, MANAGERS repeated 18 times, and DIRECTORS listed twice.

This input file can be built using an editor or The Generator. The value in the Repeat column can be used to easily generate this file.

When the file is generated at run-time the distribution will reflect the input file distribution. The larger the input file the finer the degree of granularity. This approach can be used for any combination of values.

HOW TO USE SEED VALUES, STRINGS AND INPUT FILES

Certain fill-types can have a seed value. This will create a unique start point for the random number generator. As long as the same file and field specs are entered, this value ensures a randomly generated file can be re-generated with the exact output as before. The Seed value can be in a range of 1-9999.

The STRIN Data-Fill type can contain any combination of characters and numbers, symbols and punctuation. The STRIN can also be used to create custom delimiters in special cases. Simply place the delimiter before and after the field by creating fields of single width and using STRIN to denote the value.

If the user selects CUSTA, CUSTN, or READF, the associated input source file name is entered here. This file will be read at run-time.

HOW TO HANDLE REFERENTIAL INTEGRITY

The Generator provides support for referential integrity through several techniques.

To create a column or field that will exhibit unique characteristics, by using the Unique sequence option, The Generator will produce random but unique keys.

By insuring that the number of records and seed value are the same, the unique sequence of numbers can be regenerated for files that contain primary or foreign keys.

Non-numeric keys must be provided by the user in the form of a CUSTA or CUSTN file. The Generator can generate keys from these files by reading the files sequentially and insuring that the number of records to be generated, does not exceed the number of unique values to be read in to the output file.

By writing extraction scripts in SQL, COBOL or another language, the user can create multiple output specifications from a single script. By detecting primary and foreign keys, the specifications can contain the values to maintain referential integrity.

HOW TO GENERATE NULLS AND BLANKS

There are times when a user would like a column to be blank or random records/fields to be blank.

To generate a blank column use the BLANK data -fill type. This will create a blank space for the particular column or field.

To generate random blank or NULL values, depending upon the implementation of the data type in the application, the user can randomly place blank lines or the value NULL in the input file. The more instances of a blank or NULL value occur in the file, the greater the occurrence of these values in the output file.

HOW TO USE BATCH FILES

The Generator has an open architecture. Any application , DBMS, or programming language can be   interfaced with the engine. The batch program can be written to extract file and record structure from any   program and reformat these values into a spec file that The Generator can read. This includes
specification files written on any platform or operating system. Sample batch programs are provided in the   user notes.

HOW TO USE COMMAND LINE GENERATION

The Generator can also be run on most platforms from the command line.

The command line argument to run The Generator is C:\GENERATE> DBG0 GENSPEC.TXT

(The character after DBG is the number ZERO not the letter O)

This provides the user with a very quick method for generating an output file and then editing the spec directly to make changes, and then re-running The Generator.

HOW TO ACCESS ADVANCED DATA-TYPES

The data-types for The Generator are being continually expanded based on user feedback. Some of the data-types being developed are included in the list below. Contact Mach One for availability and any recommendations you may have regarding new data-types.

Name	Description
STAMP	DB2 timestamp
JULAN	Julian dates
PACKD	Packed Decimal
ZONED	Zoned Decimal
HEXAD	Hexadecimal
BINRY	Binary
RANDM	Random Alpha-numeric strings
CALCU	Calculated field, add the value of one field to another
PADED	Intergers padded with leading zero
BOOLE	IF-THEN-ELSE Field Masks
DBG1	The executable to generate mixed record structures within the same file.

Extended data-fill types are exist for international date and time formats including masked and un-masked output. These data-fill types are supported only in the advanced versions of The Generator and may be operating system dependent.

HOW TO USE THE SEED VALUE

The Seed Value field is used to seed the random sequence of values generated. The Seed can be any number from 1 through 999,999. The number selected can be used again to produce the same sequence of values for any subsequent generation.

HOW TO GENERATE UNIQUE VALUES

Unique is a selection under the Order Type option in the Field Spec. It is only valid for the INTER data-fill type. By selecting Unique, INTER, a number of output records, and lower and upper bounds, the Generator first creates a reference file of all possible values. This file is referenced during generation to insure unique values that can be used as keys.

NOTE: Insure you have enough hard disk space to accommodate the reference file.

HOW TO GENERATE KEYS

In many database files a number of fields can be designated as KEYS for indexes. Keys can be made up of anything the user decides. The most important consideration is that the values generated be unique where the application requires this. The best way to build Unique Keys is to build a separate spec, using NULL as the field separator, and build the key a segment at a time. For example, combining INTER, NAMEL, and PHOAC would build a KEY that contains a sequential number, part of the last name, and the Area Code.

HOW TO HANDLE PRIMARY AND FOREIGN KEYS

In relational databases,   it is important to maintain relationships between tables.   One way to do this is to insure that the values in one table column define the values in another table column.   As long as the Seed value, # Records, and any bounds are kept the same,   the values will be the same.   In most cases,   the user will provide an input file using the CUSTA data-fill type to insure the same values are used.
Remember,   a primary key can only be generated for as many records as there are unique occurrences of the value. For example,   a table using days of the week as a primary key can only contain 7 records, one for each day.

HOW TO USE THE REPEAT VALUE

Repeat is a useful function for grouping values together in a parent - child relationship.   For example,   if you are creating a customer database,   and each customer should have 10 invoice numbers assigned to him or her, set the repeat value in the NAMEL field to 10.   The next field will go through 10 generations before the name changes again.

HOW TO CREATE CUSTOM DELIMITERS

The user can create custom delimiters by using the STRIN data-fill type and placing the desired delimiter in field 12 of the field spec. This STRIN data-fill type should be placed between each field in the spec. The field delimiter used in this case should be NULL.

HOW TO GENERATE LARGE NUMBERS OR DECIMAL VALUES

There may be occasions when large numbers or numbers with large decimal places can be generated. The default field width for INTER is 9. The largest decimal place is 6. To generate any large number, use INTER to set the positive side of a number, follow INTER with a STRIN and enter a period, and then follow STRIN with another INTER to generate the decimal numbers. You can use FIXED or VARIABLE record types to modify the justification of the output.

HOW TO GENERATE LARGE ALPHA-NUMERIC SEQUENCES

There are times when you may want to generate output that contains sequences of alpha and numeric values. Since these sequences will have order dependencies, you should first create a separate file spec to build these field values. CUSTA can be used to read the sequence of alpha characters for a given field. INTER can be used to build the sequence of numeric numbers. Lets take the customer account number below.

In the case below, a bank needs to create internal accounting tracking numbers for its projected 300,000 new debit card customers next year. The value looks like an alpha-numeric string of letters and numbers CCCNNNNNNCNNN as described below.

First Three Characters can be NYA, NYB, or NYC. Create a CUSTA file with these three values. Since you need 100,000 occurrences of each, set the repeat value to 100000.

The next 6 characters maybe a customer number starting at 510,000, skipping 5 numbers and generating another value. Use upper and lower bounds to set the limits and interval set to 5 to generate the next number.

The next character may be a placeholder character X. Use STRIN to insert the X.

Finally, the last 3 digits may be a number between 100 and 200. Use INTER again to generate this number. The output of this generation will be a mixed alpha-numeric sequence according to the spec file entries. This file output can then be used as in input or CUSTA file to a larger spec file to be generated.

HOW TO LINK VALUES BETWEEN FIELDS

There are many times when the values in each field of a record need to appear in the same record of the output file. READF is the data-fill type used to accomplish this task.

For example, an output file must contain a related sequences of codes for customer transactions. The codes are described in a text file as follows;

```
DAY   8-12  ATM
DAY   12-4  ATM
DAY   4-8   ATM
EVE   8-12  ATM
EVE   12-4  ATM
EVE   4-8   ATM
```

The offsets and lengths are illustrated for each field below.

(Offsets	0	5	12)
(Length	3	4	3)

These values are stored in a file that is read in at run-time. Every time a field is read from a field using the offset and length, the same record will be referenced for any other field in that record. As long as the READF data-fill type is used and the input file is referenced using the offset and length, the data will be linked at generation time.

In field 12 of the spec file, in the field spec section, enter the file name, offset, and length. For example, CODES.TXT,0,3 would read in field 1, the DAY or EVE values.

USER NOTES

The Generator is a software tool that is modular in design. This allows different components to run on different platforms and still be used together.

The engine can reside on the mainframe, users can be using DOS, Windows, or UNIX on their workstations and ship the spec file to the mainframe fro generation.

The Generator is designed for team integration. A team leader can be responsible for creating all the user-defined input files that everyone else will use. This allows for unit testing and concurrency testing across the development project.

The engine can reside on a server, providing easy access to all users on the network.

Custom programs can be written to access and read data file structures and preformat spec files. For example, an SQL program can be written to read all the system tables and automatically create spec files for every table in the application.

A user could create an XBASE front end to build spec files and then generate spec files in the form of a report from the report program.

The Generator engine can be distributed (under separate license agreement) with a spec file to users of an application without having to ship large data files, saving disk space.

These are just a few examples of how the modular design can be adapted to your environment.

MAINFRAME USERS

The MVS version of The Generator can be run in batch mode or from a ISPF Dialog provided with the product. Sample installation JCL and execution JCL is provided with the tape. Under MVS many of the file, record and field limits are large than the workstation versions of The Generator.

The VM/CMS version of The Generator must be run in batch mode. Sample installation JCL and execution JCL is provided with the tape. Under VM/CMS many of the file, record and field limits are large than the workstation versions of The Generator.

Users can create the specification on the workstation and upload the spec to the mainframe, if the mainframe version of the engine is resident on the CPU.

SQL USERS

Because of the open architecture of The Generator, use with the SQL language will be dependent on the version of SQL in use at the installation.

A simple SQL procedure can be run against the system tables to SELECT and PRINT the required field information for a specification. Once standardized SQL exists, the user only needs to modify the TABLE name to create another specification. Sample SQL is provided with the installation tape, cartridge, or disk.

UNIX USERS

The UNIX version must be run from the command-line.   Most UNIX DBMSs utilize SQL and the user may want to refer to the SQL section for more information. Otherwise, the installation instructions provided with the disk, tape, or cartridge will have updates   regarding the available GUIs for UNIX users.

AS/400 USERS

The Generator can be run as a called program from within an RPG, COBOL, or C/400 program. The engine resides on the AS/400 host and users can create and edit specifications on the host or attached workstations. SQL support is also an option for those shops using SQL.

DOS AND OS/2 USERS

A DOS version of The Generator is available, and can be run from the command line, batch, or from the GUI.
The installation diskette includes instructions concerning activation of the user interface. The workstation version can also be integrated with any other 3rd party DBMS, language, or case tool.
The Windows version will run as a task under OS/2. Contact Mach One Software for native OS/2 release date

EDUCATIONAL USERS

One of the most popular uses of The Generator is in the classroom.   Students can purchase The Generator at a special educational discount through the college or university bookstore.   Group discounts for classes are also available for teachers by contacting Mach One directly.

ERROR MESSAGES

Error messages originate from two sources.

The first area to check is the Spec file,   insuring all the fields have been completed.   This occurs most often when reading a DB header and not completing the spec.

The other area to check is the Drive and Sub-directory where the spec file was saved.   The Spec File must be saved in the same DRIVE and SUB-DIRECTORY where the Generator program files reside.

To output internal Generator error messages follow these steps.
1) Exit Windows
2) Change to the Generator directory
3) Enter DBG0 TheSpecFileName >error.log
4) Example: DBG0 SAMPLE.SPF >error.log

This will write any error messages not displayed in Windows to the file error.log.   Read the error.log file with any text editor to find out what the message says and correct the error.