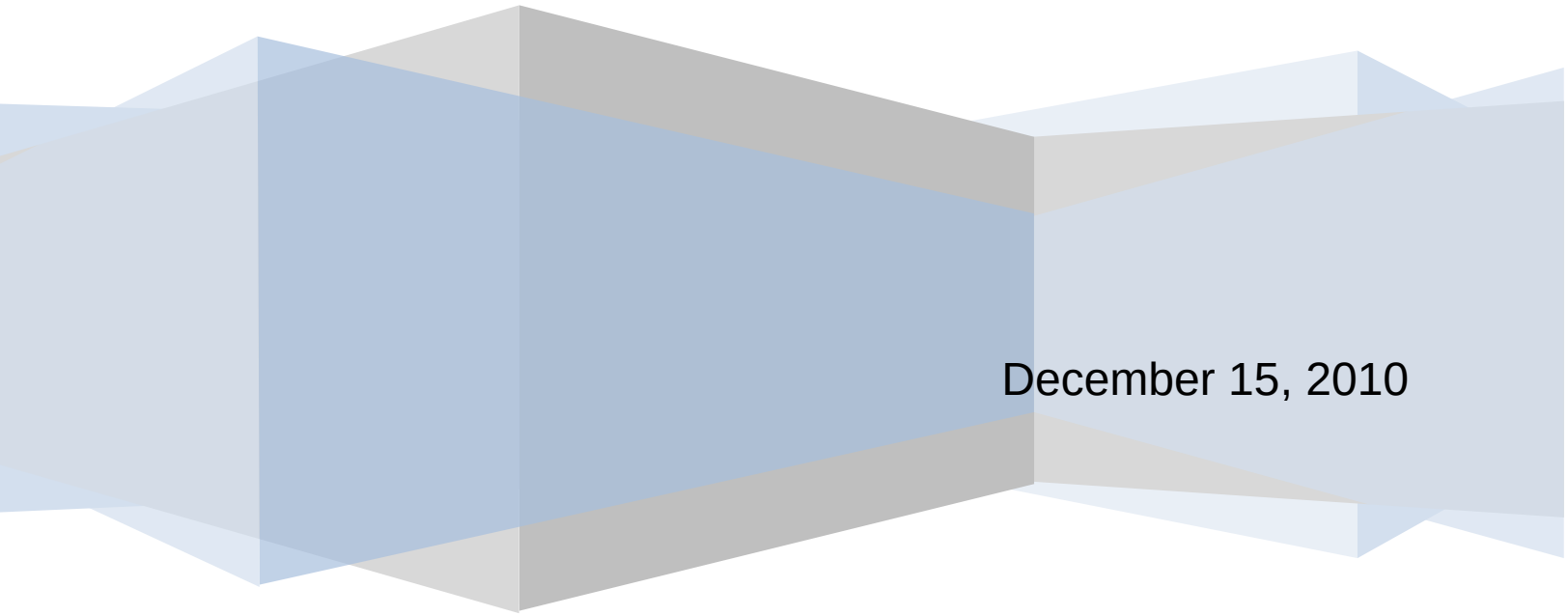
Gendatarator Freeform Users Manual (Ver. 2.4)

Generating your data, your way.

Glenn Thomas



December 15, 2010

Cydonian Technologies, LLC

Contents

Copyright 2010 Cydonian Technologies, LLC

Conventions used in this Document

Wherever possible, standard conventions are used to identify differences between commands, data, and directions.

Typed Commands – Whenever you are directed to type/enter text (or select an entry from a dropdown selector), the text will be displayed as:

> ***Example of typed text***

Menu Commands – If you are directed to traverse a series of menu options/dropdown selectors, the directions will be displayed as:

> **Address** > **City** > **Insert Button**

In the above example, you would click “Address”, then “City”, and then click the “Insert Button”.

Data Output – Since the purpose of Gendatarator is to generate data, output from the program will be displayed and referenced frequently. When output data is displayed, it will appear in this format:

```
7641-32,"Murphy","Lorna","1842 McCaan Quay Smt","Folk","MO","65085"
7845-65,"Arias","Reed","5514 Parkview Ct","Stratford","WA","98853"
5767-36,"Owen","Marlin","7766 Bernadotte Pike","Alsace Manor","PA","19560"
```

To start using Gendatarator simply go to www.gendatarator.com and click the “Freeform” tab.

Copyright 2010 Cydonian Technologies, LLC

What is Gendatarator?

Gendatarator (Gen-data-rator) is a tool that is designed and developed to solve common problems encountered by software developers and testers. Quite simply, Gendatarator allows the generation of complete sets of test data for use in loading databases, testing programs, or just providing data that has a real-world look and feel.

Generally, the creation of valid test data can be almost as involved as the actual development of the program code itself. A large number of hours are spent writing scripts that will be used a limited number of times to create data to meet the criteria for testing databases and programs. As a result, many organizations use archival or old data from a backup - which may not fully test new applications. The use of archival data may also create security concerns if it contains sensitive information such as credit card numbers or SSN's. Think of sending an archival list of sensitive customer information to a consultant, much less an offshore developer!!!

Gendatarator was created to allow organizations to quickly, easily, and effectively create real-world test data. It requires no coding or programming knowledge – you only need to understand the format for the data you wish to create. Gendatarator allows complete control over how data is generated, and it allows format changes as often as your requirements dictate.

Here is a quick example of Gendatarator in a real-world case:

You need a set of test data for a customer database that includes a customer number, last name, first name, street address, city, state, and a zip code in the following format –

7641-32,"Murphy","Lorna","1842 McCaan Quay Smt","Folk","MO","65085"
7845-65,"Arias","Reed","5514 Parkview Ct","Stratford","WA","98853"
5767-36,"Owen","Marlin","7766 Bernadotte Pike","Alsace Manor","PA","19560"

You would simply use the onscreen tools to compose the following text string in the "Token List" text box (either directly or via the dropdown selectors) –

9000\-00,"[Last]","[First]","[Address1]","[City]","[STATE]","[zip]"

Click "Output Format" > "HTML"

Click 

…and view your data.

It really is that easy!

Gendatarator creates test data based on the ***tokens***. Tokens represent the smallest pieces of data that can be generated. In the example above, "[First]" is a token, as is "9". Gendatarator supports a wide range of tokens and token combinations, and more are in development. Many tokens also support formatting options such as UPPERCASE, lowercase, Capitalized, or embedded characters. For instance, "[FIRST]" will generate a name in the format "FRED" or "BARNEY" and "[last]" will generate a last name in the format "smith". The "[SSN]" token will generate a nine digit Social Security Number (no spaces or dashes), whereas "[SSN-]" would generate a SSN with embedded dashes (e.g. "123-45-6789".)

A complete list of tokens and formatting options is available as the "Gendatarator Token Master List" or "Cheatsheet" which is available on the website.

Copyright 2010 Cydonian Technologies, LLC

Why is Gendatarator free?

Gendatarator Freeform is the first of three products that is planned for release. Two more products are in development that will have enhanced capabilities and features. Gendatarator Plus and Gendatarator Pro will expand the capabilities to associate a set of tokens with a specific column/field in a database. Both of these products will also allow you to save your token lists online as "Generators". Gendatarator Pro will significantly expand the number of generators (e.g. sets of data) you can save and also the number of rows you can generate. Gendatarator Pro is designed for professionals who need to generate larger quantities of data or who do frequent testing.

Here is a quick overview of the planned products:

	Gendatarator	*Gendatarator Plus*	*Gendatarator Pro*
Cost	*FREE*	*Free*	*Subscription (Cost TBD)*
Available Date	*NOW*	*TBD*	*TBD*
Max Number of rows generated (per run)	*250*	*500*	*5000*
Ability to save token lists (Generators)?	*No*	*Yes (with registration)*	*Yes (with subscription)*
Number of saved token lists (Generators).	*N/A*	*5*	*250*
Output Formats	*CSV, HTML, SQL*	*CSV, HTML, SQL, XML*	*CSV, HTML, SQL, XML*

Copyright 2010 Cydonian Technologies, LLC

How to use Gendatarator

Gendatarator was designed to be easy to use and intuitive. To create test data, you simply select from the set of available tokens that represent the data that you need, and click “Generate Data”. You can enter tokens either by typing them manually, or by using the dropdown selectors. Obviously, the most import thing to understand is what type of data you need to create and how it should be represented. A simple field like “price” in a database may have constraints placed on it for size, number of decimal places, and embedded commas or a dollar sign. If you do not understand what you need to generate, it may impact the quality of the data you create.

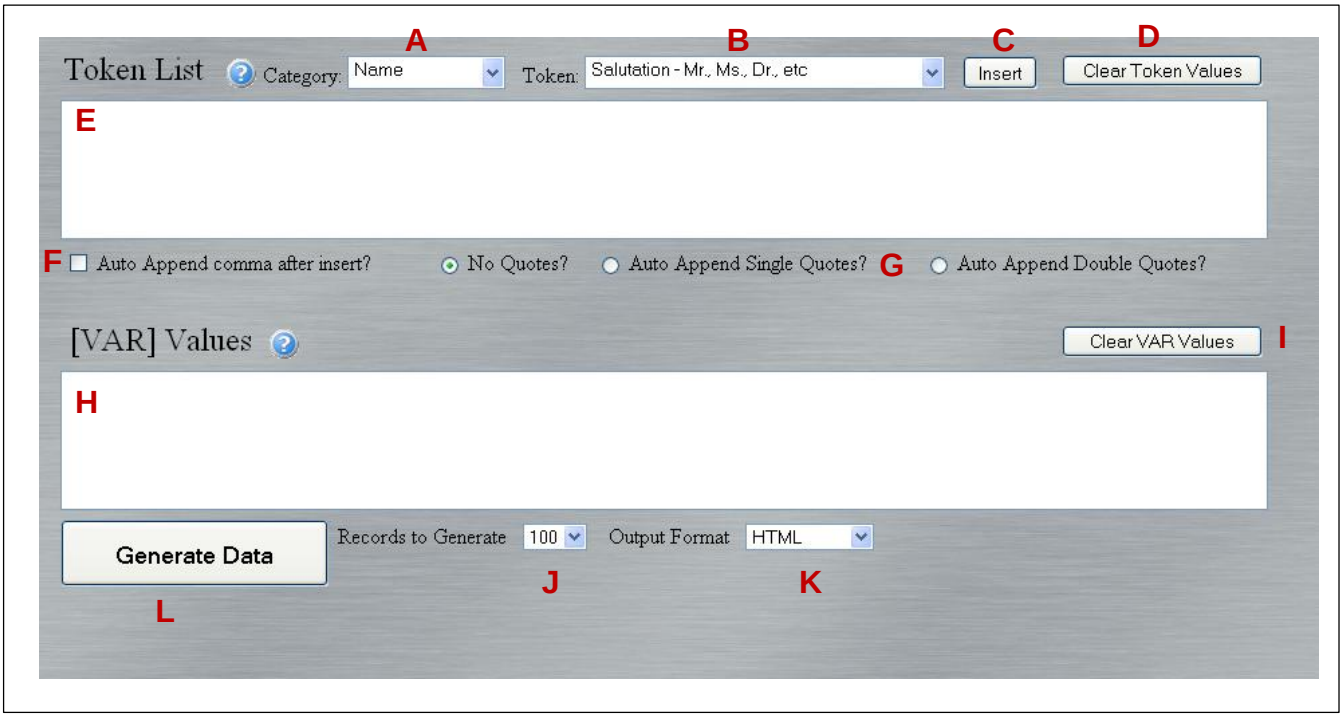
Here are many of the “canned” values that Gendatarator can generate.

Characters	Alphabetic, numeric, uppercase, lowercase, hexadecimal, punctuation, and any combination of these
Names	Male, female, generic, last, and middle initial
Addresses	Street address, city, state, zip code, country, area code, phone number
Dates	Year (2 or 4 digit) month name/number, day name/number, Julian
Time	12/24 hours, minutes, seconds, milliseconds, AM/PM
Numeric values	General, integer, floating, money, incremental values, variable size (such as 2-4 digits), zero-padded
Miscellaneous	Words, SSN’s, IP addresses, colors, domains, email addresses, credit card numbers, and more
Custom	Build your own tokens by combining existing tokens, or building you own sets of values
Literals	Embed any characters or strings of characters within your data

(Refer to the Gendatarator Token Master List (Appendix A) for a complete list of the tokens and their formatting.)

A tour around the screen.

Before we begin actually creating data, let’s take a quick look at the Gendatarator screen.



Copyright 2010 Cydonian Technologies, LLC

A – Category Selector. This will allow you to choose the type or ***category*** of token you need. Examples of categories include Name, Address, Number, and Date. There is also a Common category which includes the tokens that are used most frequently. Selecting a category will populate the Token Selector with the appropriate token values.

B – Token Selector. This allows you to select the actual token you wish to use. Depending on the selected category, different choices for tokens will be displayed. For instance, choosing the Address category will allow token choices such as “City”, “State (Full Name)” and “Zip code -5 digit”. Selecting a token will not automatically populate the Token List Textbox (you must click the Insert Button).

C – Insert Button. Clicking the insert button will insert the selected token into the Token List Textbox ***at the cursor position***. Clicking it repeatedly will insert multiple values.

D – Clear Token Values Button. Clicking this button will clear all values in the Token List Textbox.

E – Token List Textbox. Values can be entered into this textbox either through the category/token dropdowns or by typing directly into the textbox. Cut-and-paste is fully supported, so token lists can be composed and saved using any text tool.

F – Auto Append Comma. If checked, Gendatarator will automatically append a comma after each token when the Insert Button is clicked.

G – Quotes. Depending on the selected radio button, Gendatarator will automatically place either single or double quotes around the token when the Insert Button is clicked. The default is “No Quotes”. If “Auto Append Comma” is checked, the comma is placed after any quotes.

H – [VAR] Values Textbox. This textbox allows you to create your own lists of custom values to substitute for a token. For instance, if you need a token that represents departments in a company, you would enter the values in the [VAR] Values Textbox as “Accounting,IT,Marketing,Sales” (i.e. values separated by commas). Placing the value “[Var]” in the Token List Textbox will cause these values to be substituted when the data is generated.

I – Clear VAR Values Button. Clicking this button will clear all values in the VAR Values Textbox.

J – Records to Generate Selector. This allows you to select the number of records (rows) that will be generated.

K – Output Format Selector. Use this to choose the format for the output of the data that is being generated. Currently, available formats include HTML, SQL, and Comma-Separated Values (CSV).

L – Generate Data Button. Clicking this button will initiate the creation of test data according to the values in the token list, var values list, and the other selections on the screen.

Copyright 2010 Cydonian Technologies, LLC

Generating your first data.

Now that we know our way around the screen, let's create a simple set of test data from scratch. In this example, we will create a comma-separated list of values based on the same token list that we used in the first example.

7641-32,"Murphy","Lorna","1842 McCaan Quay Smt","Folk","MO","65085"

For this example, it must match the following format:

customer number (4 digits with a dash followed by two digits – cannot have leading zeros),
last name (in quotes),
first name (in quotes) ,
street address (in quotes),
city (in quotes),
state (in quotes),
zip code (5 digits in quotes)

First, let's create the customer number. The token '9' will generate any digit 1 through 9, and the token '0' will generate any digit 0 through 9. This is important to know if you need to make sure that leading zeros are not generated! So place the cursor in the Token List Textbox, and enter the value '9000' to create the first part of our customer number. Next, we want to create a dash character. To use any character as a ***literal***, we simply precede the character with the '\' token. In this case, we would enter '\-'. Next, we add '00' to generate the last two digits. For our example, the last two digits can contain zeros. Finally, we add a comma to separate it from the next value. Our token list now looks like this - ***'9000\-00,'***

Notice that we need to enclose all of our remaining fields in double quotes and separate them with commas. To make the entry easier, click the "Append Double Quotes" radio button, and click the "Auto Append comma after insert" checkbox.

Now just click the following (in order)

Category> Name	**Token> Last Name (Static)**	**>Insert Button**	**(inserts "[Last]",)**
Category> Name	**Token> First Name (Static) - Generic**	**>Insert Button**	**(inserts "[First]",)**
Category> Address	**Token> Address – 1234-A Oak St**	**>Insert Button**	**(inserts "[Address1]",)**
Category> Address	**Token> City**	**>Insert Button**	**(inserts "[City]",)**
Category> Address	**Token> State (2 character abbreviation)**	**>Insert Button**	**(inserts "[STATE]",)**
Category> Address	**Token> Zip code (5 digit)**	**>Insert Button**	**(inserts "[zip]",)**

Delete the last comma that was inserted at the end of the string

The string in the token list textbox should now look like this:

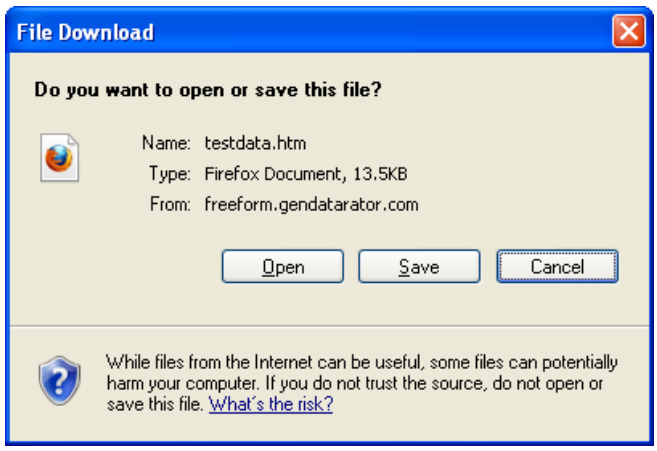
9000\-00,"[Last]","[First]","[Address1]","[City]","[STATE]","[zip]"

To see the output, perform the following:

Output Format> HTML >Generate Data Button.

Copyright 2010 Cydonian Technologies, LLC

If your system is configured correctly, you should see a popup that looks like one of the following that will ask if you wish to open or save the data. Which message you see will depend on the browser you are using (IE or Firefox). ***Click "open" or "OK"***:



You should see a list of data that appears in a browser window that looks like the following:

7636-52,"Coffey","Angela","6820-B Gannet Park","Zenoria","LA","71371"
8722-49,"Delgado","Winfred","9713 Riddle Rd","Conley","KY","41465"
7464-05,"Fox","Madalene","1039 Hydenwood Trce","Forest Grove","MN","56660"
7915-56,"Clements","Neville","3463-W Swordfish Trce","Winston-Salem","NC","27127"
Etc.

It's that simple!

(If you were not successful in creating the example data, please consult "Appendix B – Troubleshooting" in this document)

Static vs. Dynamic Tokens

As you explore the Freeform screen, you may notice two words that are used with some tokens – ***Static*** and ***Dynamic***. These refer to whether the generated value is constant for that row of data. The best way to illustrate this is with an example:

If we generate data using the 'First Name (Static) - Male' selection, it will insert the '[Male]' token. If we use the following token list - '[Male], [Male], [Male]' – it will create data in the following format:

Fred,Fred,Fred
Larry,Larry,Larry
Ralph,Ralph,Ralph

However, if we generate data using the 'Random Male Name' selection, it will insert the '[Randommale]' token. Using the token list ''[Randommale], '[Randommale], '[Randommale]' which is composed of ***dynamic*** tokens will create data in the following format:

Dino,Jacob,Dorian
Mark,Sammy,Kraig
Chance,Jamison,Solomon

Copyright 2010 Cydonian Technologies, LLC

Why use one over the other*?** In most cases it will not matter. This option was created to allow users to ***re-use a value within a row of data. A good example would be using the last name as part of the email address in a generation of data. Address information, and some date tokens are other examples of dynamic tokens. There may be cases where data such as a date or address may need to be used more than once in a row with the same value.

More Examples

Now that you know your way around the screen, let's look at some more examples of creating data.

Example 1

Using our last example, let's substitute a region code for the last two digits in the customer number. Valid regions are "NW, SW, SE, and NW". Simply enter "NW,NE,SW,SE" into the VAR Values Textbox, and change the Token List string to look like the following:

9000\-[VAR],"[Last]","[First]","[Address1]","[City]","[STATE]","[zip]"

When you click the Generate Data Button, you should see the new output:

4193-SE,"Alexander","Norberto","8057 Fernbridge Smt","Madison","ME","4950"
6801-NE,"Harper","Vivian","8120 Whitley Abbey Blvd","Imlaystown","NJ","8526"
7388-NW,"Wolfe","Mellissa","1046 Charity Neck Ave","Drury","MA","1343"

Remember: Many tokens are case sensitive! Using [VAR] will generate a value such as "NW", but [var] will generate "nw".

If you want to increase the selection of a given value (e.g. half of the records should have "NW" as a region code), simply enter the value multiple times in the VAR Values Textbox, such as "NW,NW,NW,NE,SW,SE".

Copyright 2010 Cydonian Technologies, LLC

Example 2

Joe is creating test data for his inventory table. One of the key fields is “Part Number” that typically looks like this:

SE-P47123-W

While the database might accept any character string, the data might need to conform to a specific format to test new programs that are being written. Joe finds out that the part number data is composed of the following parts:

SE = Region code (SE, NW, SW, or NE)
P = Product Type code (P, E, C, X, or Q)
47123 = Product number (exactly 5 digit integer – leading zeros if necessary)
W = Product color code (“W”hite, “R”ed, ”B”lue, blac”K”, “G”reen)

Now that Joe knows the structure for the data, he can use the tokens that are available in Gendatarator to compose the field. Since much of his format consists of “sets” of data (such as a region code in the set - SE, NW, SW, or NE), he selects the ***Set token*** “(S||)” to represent parts of his token. The Set token type allows the creation of custom sets to supply a value, and operates similar to the [VAR] token.

The format for the Set token is:
(S|val1|val2|val3|etc…|)

Joe created the following token list meet his needs:

(S|NW|SW|NE|SE|)-(S|P|E|C|X|Q|)00000\-(S|W|R|B|K|G|)

And the following data is generated:

SW-C28943-K
NE-E67282-B
SW-Q82945-K
Etc…

Copyright 2010 Cydonian Technologies, LLC

Example 3

This example is a bit more complex and introduces several new types of tokens – “Incremental”, “Literal”, “Date”, and “VariSize”.

Incremental tokens allow you to create incremental values such as “1,2,3,4,5,…” or “0100,0107,0114,0121,…”. You can also fix the number of digits and or precede it with a zero or other character. The format for the incremental token is:

(I,staring_val,increment_value,number_of_digits,pad_character)

Literal tokens will place a static value in each record/row. We used a literal token in a previous example when we used the value “\-“ to place a dash at a specific location in the output data. When there are longer strings of literal data, it can be enclosed in braces “{}” rather than preceding each character with a “\”. For instance, “{This is a test}” is the same as “\T\h\i\s\ \i\s\ \a\ \t\e\s\t” and will place the string into the output data.

Date tokens are fairly straight forward and are referenced quite well on the “Gendatarator Master Token List”. The only complex aspect is that the “[initdate]” token will initialize the generation of a date to fall within a range of values. For instance, [initdate,1/1/1970,1/1/1980] will require that all following date tokens generate a date within the defined range of 1/1/1970-1/1/1980. The [initdate] token can be used multiple times in a token list if you need to use several date ranges. The [initdate] token does not need to be formatted with any commas or quotes.

Lastly, the “VariSize” token allows you to create a string of characters of a variable length. For instance, you may need to generate 4 to 6 uppercase characters or 3 to 8 hexadecimal characters. VariSize use the format:

(char_type,min_size,max_size)

Joe is now creating test data for his customer transaction table. The columns are as follows (all fields in quotes and comma-separated):

Account Number (4 digit current year, a dash, a sequence number of exactly 5 digits 10001-99999)
Customer First Name,
Customer Last Name,
Email address,
Birth date, YYYY-MM-DD (valid range is 1/1/1920-1/1/2010)
site password (6-18 characters including upper, lower, numeric, and punctuation)

Referring to the “Gendatarator Master Token List” we would define the following tokens:

*account Number - **{2010}\-(I,10001,1,5,0)***
*customer First Name - **“[First]”,***
*customer Last Name - **“[Last]”,***
*email address - **“[email]”,***
*birth date - **[initdate,1/1/1920,1/1/2010]"[Date4]",***
*site password - **"(U,6,18)"***

And the complete token string is as follows:

“{2010}\-(I,10001,1,5,0)","[First]","[Last]","[email]", [initdate,1/1/1920,1/1/2010]"[Date4]","(U,6,18)"

And the following data is generated:

"2010-10001","Angeline","Sherman","xylplw8m92u@vrow6pfvpv.int", "1954-04-06","*B]^$(ZXg-GL"
"2010-10002","Lashawn","Lowery","svnsx6wx8@91obat6n6jy.com", "1984-10-02","i:H2*YX"
"2010-10003","Keli","Mayo","6rwys1@1bpcn5lvp.gov", "1977-02-22","x4c)Cj}4}WX><nJ1{y"

Copyright 2010 Cydonian Technologies, LLC

Frequently and Infrequently Asked Questions (FAIAQ's)

Please check the website for the most current FAQ's

The Future of Gendatarator

We have big plans for adding capabilities and expanding the number of tokens. Please stay in touch with new developments through the "Latest News" area on the Home page of our website.

Copyright 2010 Cydonian Technologies, LLC

Appendix A - Gendatarator Token Master List

For the most current and highest quality copy of the Gendatarator Master Token List, please visit:

http://www.gendatarator.com

Copyright 2010 Cydonian Technologies, LLC

Appendix B - Terminology

CSV - Comma Separated Values. A structure of data that uses commas as a separator between the data fields.

Dynamic Value – A generated value that WILL NOT remain constant within a row of data. For instance, “[Randomfirst] - [Randomfirst]” would generate a value such as “Allen – Sally”.

Static Value – A generated value that WILL remain constant within a row of data. For instance, “[First] - [First]” would generate a value such as “Barbara – Barbara”.

Token – an element within Gendatarator that is responsible for generating a unique piece of data, such as a name, character, number, address, or color.

Token List – a string of tokens that makes up a request to generate one or more rows of data.

[VAR] – The VARiable textbox. Any comma-separated values that are placed in this textbox will be selected at random and substituted in the Token List Textbox wherever the [VAR] value is used. The [VAR] value can be represented as “[VAR]”, “[Var]”, or “[var]”, depending on the desired capitalization requirements.

Copyright 2010 Cydonian Technologies, LLC

Appendix C - Troubleshooting

Basics –

1) **I do not get a browser popup that allows me to view the output.** First, check to make sure that "HTML" is the Output Format. Next, ensure that the file type ".htm" is associated with your browser.

2) **I got an error message –"ERROR - Unidentified token 'X' at position Y". How do I fix this?** Look at that position in your token list string and ensure that it is a valid character token, or that the token at that position is in a valid format. Common problems include an invalid format for a more complex token (such as an Incremental token) or a token that has lost a character due to a cut-and-paste action. If you are still unsure, remove the tokens from that point on and recompose the token list.

Please refer to the troubleshooting section of the website under "Documentation" for the most current information.

Copyright 2010 Cydonian Technologies, LLC