**Generic CSV Generator**

Author: Brendan More O’Ferrall ([mooreof@ie.ibm.com](mailto:mooreof@ie.ibm.com))

Date: 3rd November 2013

**About**

This document describes a (Perl) script that can be used to generate custom CSV files that can be imported into any user-defined table (was initially written to test performance of the data ingestion feature used by Intelligent Operations Center – IOC - 1.6). You can specify any number of columns with any name and one of the following column definition types: SEQUENCE, INTEGER, FLOAT, DATETIME, DATE, TIME, ENUM, POINT, LINESTRING, MULTILINESTRING, POLYGON, STRNUM, STRSEQ, PERL, STRING.

**Installation**

To install on RHEL, log in as root and copy the attached Perl and sample configuration files to any directory.

Convert the files to UNIX format:  
> dos2unix generic\_csv\_writer.\*  
Make the perl script executable:  
> chmod +x generic\_csv\_writer.pl

There is a dependency on the DateTime module which isn't installed by default on RHEL Perl distributions.  
Use the cpan program to install DateTime.  
At the command prompt start the cpan program:  
> cpan  
  
If this is the first time using cpan you will be prompted to initialize CPAN's configuration. Choose the auto-configure option for convenience.  
In order to avoid constant prompting during module installation exit from cpan ('quit') and edit the configuration file:

/usr/share/perl5/CPAN/Config.pm (RHEL)  
  
change 'ask' -> 'follow' in the following line:

 'prerequisites\_policy' => q[follow],  
  
save the file, then restart cpan.  
Run the following commands to install the necessary modules:

*RHEL6 (all IOC16-based environments):*  
cpan> force install Module::Build  
cpan> force install DateTime

*RHEL5 (all IOC15-based environments):*  
cpan> force install Module::Build  
cpan> force install DateTime  
cpan> force install Params::Validate  
cpan> force install Module::Implementation  
cpan> force install DateTime::Locale  
cpan> force install DateTime::TimeZone

*Windows*:  
A distribution such as ActivePerl or Strawberry can be used. Use ppm or cpan to install missing modules.

*AIX*:  
For AIX systems cpan might not work. Try the approach described here: https://www.ibm.com/developerworks/community/blogs/Dougclectica/entry/perl\_modules\_and\_aix5?lang=en

If you chose to leave the cpan configuration as default ('prerequisites\_policy'=='ask'), you will be prompted several times during the install process as follows:

Shall I follow them and prepend them to the queue of modules we are processing right now? [yes]

Hit return in all cases to accept the default (yes). Finally, exit from cpan:

cpan> quit

The program should be ready for use.

**Description**

Run the program as follows:

> generic\_csv\_writer.pl -i <ini\_file>

All options are specified in the configuration file <ini\_file> (sample ini included). There are two sections in the configuration file, [OPTIONS] (various global options) and [COLUMNS] (column definitions).

**Options**

Following are the current options supported:

> base path + file name (use %d to add sequential number to file name, %t to add current date-time)  
filename=ds\_input\_%d.csv

> destination folder, output csv file is copied here (if specified)  
dest\_folder=/opt/IBM/ioc/csv

> delimiter between columns  
csv\_delimiter=,

> number of CSV files to generate  
iterations=2

> number of records per CSV file  
records\_per\_file=100

> gap in seconds between the generation of each CSV file  
> this is the exact time between the start of each iteration (i.e., it takes the time to write the CSV into account)  
secs\_between\_iterations=60

> Format to use when writing out DATETIME value to CSV file  
datetime\_format=%Y-%m-%d %H:%M:%S.000

> Format to use when writing out DATE value to CSV file  
date\_format=%Y-%m-%d

> Format to use when writing out TIME value to CSV file  
time\_format=%H:%M:%S.000

> Wrap string values in quotes (non-zero=true, 0=false)  
quote\_string\_values=0

> Character to separate list of enumerated values for type ENUM  
enum\_separator=,

> Display format for type FLOAT  
float\_format=%.6f

**Column definitions**

Each column definition is specified as follows:

COLNAME=COLUMN SPECIFICATION

where COLNAME is the name of the column and COLUMN SPECIFICATION is a specification that will vary depending on the data type chosen. The following data types are supported:

***SEQUENCE***

COLNAME=SEQUENCE;STARTVALUE;INCREMENTSIZE;ZEROPAD SPECIFICATION;RESTART

where SEQUENCE identifies the data type, STARTVALUE is the initial value for the sequence, INCREMENTSIZE is the amount by which the value is incremented each row, ZEROPAD SPECIFICATION allows you to specify that the sequence is zero-padded and RESTART allows you to specify that the value reverts to STARTVALUE after RESTART records (RESTART is disabled if set to 0 or omitted). For example:

INDEXNUM=SEQUENCE;1;1;6;10

i.e., generate a column called INDEXNUM, starting with a value 1, incremented by 1 each time, zero-padded to 6 digits; the value reverts to 1 every 10 records. Note that the Sequence's value doesn't restart with each new CSV file.

***INTEGER***

COLNAME=INTEGER;RANDOM\_RANGESTART;RANDOM\_RANGEEND

where INTEGER identifies the data type, and RANDOM\_RANGESTART/RANDOM\_RANGEEND identify the range between which a random integer is generated. RANDOM\_RANGEEND is exclusive. For example:

PERMITID=INTEGER;1000;10000

i.e., generate a column called PERMITID with a value equal to a random integer between 1000 and 10000 (exclusive).

***FLOAT***

COLNAME=FLOAT;RANDOM\_RANGESTART;RANDOM\_RANGEEND

where FLOAT identifies the data type, and RANDOM\_RANGESTART/RANDOM\_RANGEEND identify the range between which a random number is generated. RANDOM\_RANGEEND is exclusive. For example:

READING=FLOAT;0;50

i.e., generate a column called READING with a value equal to a random number between 0 and 50 (exclusive).

***ENUM***

COLNAME=ENUM;LIST OF COMMA-SEPARATED VALUES

where ENUM identifies the data type, followed by a list of values separated by 'enum\_separator' (default comma). For example:

NAME=ENUM;Hurricane Approaching,Large Wildfire Moving Rapidly West,Chemical Spill Near Residential Area,Major Water Leak,Multi-vehicle Accident with Injuries

i.e., generate a column called NAME with a randomly chosen value from the list.

***DATE***

COLNAME=DATE;INITIAL DATE(mm/dd/yyyy);INCREMENT IN DAYS

where DATE identifies the data type, INITIAL DATE is the start date (must be in the format 'mm/dd/yyyy') and INCREMENT IN DAYS is the number of days to increment INITIAL DATE for each row. The format with which the value is written is specified by the value 'date\_format' in [OPTIONS] For example:

EVENTDATE=DATE;03/22/2013;1

i.e., generate a column called EVENTDATE, starting with 22nd March 2013, incrementing each subsequent value by 1 day.

***TIME***

COLNAME=TIME;INITIAL TIME(HH:MM:SS);INCREMENT IN SECONDS

where TIME identifies the data type, INITIAL TIME is the start time (must be in the format 'HH:MM:SS') and INCREMENT IN SECONDS is the number of seconds to increment INITIAL TIME for each row. The format with which the value is written is specified by the value 'time\_format' in [OPTIONS]. For example:

EVENTTIME=TIME;00:00:00;1

i.e., generate a column called EVENTTIME, starting with midnight, incrementing each subsequent value by 1 second.

***DATETIME***

COLNAME=DATETIME;INITIAL DATETIME(mm/dd/yyyy HH:MM:SS);INCREMENT IN SECONDS

where DATETIME identifies the data type, INITIAL DATETIME is the start date-time (must be in the format 'mm/dd/yyyy HH:MM:SS') and INCREMENT IN SECONDS is the number of seconds to increment INITIAL DATETIME for each row. The format with which the value is written is specified by the value 'datetime\_format' in [OPTIONS] For example:

STARTDATETIME=DATETIME;03/22/2013 00:00:00;900

i.e., generate a column called STARTDATETIME, starting with 22nd March 2013, incrementing each subsequent value by 15 minutes.

***STRSEQ***

Used to specify a combination of string characters and zero or more (optionally zero-padded) sequences.

COLNAME=STRSEQ;BASE\_VALUE(%s replaced with sequence value);START\_VALUE;INCREMENTSIZE;ZEROPAD SPECIFICATION(zero if no padding);RESTART

where STRSEQ identifies the data type, BASE\_VALUE is the base value (where all occurrences of %s are replaced with the defined sequence number), START\_VALUE is the initial value of the sequence, INCREMENTSIZE is the amount by which START\_VALUE is incremented for each row, ZEROPAD SPECIFICATION is used to define optional zero padding and RESTART allows you to specify that the value reverts to STARTVALUE after RESTART records (RESTART is disabled if set to 0 or omitted). For example:

ASSET=STRSEQ;LOC%s;1;1;6;10

i.e., generate a column called ASSET, with a value that starts with 'LOC' and ends with a sequence number that starts with 1, increments by 1 each time and is zero-padded to 6 digits; every 10 records the sequence number reverts to 1. An example output would be 'LOC000100'. Note that the Sequence's value doesn't restart with each new CSV file.

***STRNUM***

Used to specify a combination of string characters and zero or more random numbers (each one different).

COLNAME=STRNUM;BASE\_VALUE(%r replaced with random number);RANDOM\_RANGESTART;RANDOM\_RANGEEND

where STRNUM identifies the data type, BASE\_VALUE is the base value (where all occurrences of %r are replaced with different random numbers in the specified range), and RANDOM\_RANGESTART/RANDOM\_RANGEEND define the range within which the random integer is generated. For example:

ASSET=STRNUM;XXX%rYYY%rZZZ;1;100

i.e., generate a column called ASSET, with a value where each occurrence of %r is replaced with a random integer between 1 and 100. An example output would be 'XXX46YYY92ZZZ'.

***STRING***

COLNAME=VALUE

No data type qualifier, just set to the required value, e.g.:

TIMEZONEOFFSET=0

***POINT***

Used to specify a POINT value conforming to the WKT standard.

COLNAME=POINT;BASE\_LATITUDE;DELTA\_LATITUDE;BASE\_LONGITUDE;DELTA\_LONGITUDE

where POINT identifies the data type, BASE\_LATITUDE is the base latitude value, DELTA\_LATITUDE defines the upper bound within which a value is randomly generated, BASE\_LONGITUDE is the base longitude value, DELTA\_LONGITUDE defines the upper bound within which a value is randomly generated (these four values define a bounding box within which the point will be generated). For example:

LOCATION=POINT;44;1;-93;1

i.e., generate a column called LOCATION, with longitude ranging between -93 and -92 and latitude ranging between 44 and 45: e.g., "POINT(-92.182770 44.588715)".

***LINESTRING*\***

Used to specify a LINESTRING value conforming to the WKT standard.

COLNAME=LINESTRING;BASE LATITUDE,DELTA LATITUDE,BASE LONGITUDE,DELTA LONGITUDE;DELTA LENGTH;NUMBER OF POINTS

where LINESTRING identifies the data type, BASE LATITUDE is the base latitude value, DELTA\_LATITUDE defines the upper latitude bound, BASE\_LONGITUDE is the base longitude value, DELTA\_LONGITUDE defines the upper longitude bound (these four values define a bounding box with which the first points of each linestring will be generated). DELTA\_LENGTH defines the upper range between the previous point and the next point, and NUMBER\_OF\_POINTS defines the number of points generated. For example:

LOCATION2=LINESTRING;44,1,-93,1;0.1;5

will generate a column called LOCATION2, with five points: e.g., "LINESTRING(-92.456543 44.500763, -92.358408 44.596585, -92.443079 44.592947, -92.360480 44.570917, -92.433624 44.508441)".

***MULTILINESTRING*\***

Used to specify a MULTILINESTRING value conforming to the WKT standard.

COLNAME=MULTILINESTRING;BASE LATITUDE,DELTA LATITUDE,BASE LONGITUDE,DELTA LONGITUDE;DELTA LENGTH;NUMBER OF POINTS

where MULTILINESTRING identifies the data type, BASE LATITUDE is the base latitude value, DELTA\_LATITUDE defines the upper latitude bound, BASE\_LONGITUDE is the base longitude value, DELTA\_LONGITUDE defines the upper longitude bound (these four values define a bounding box with which the first points of each linestring will be generated). DELTA\_LENGTH defines the upper range between the previous point and the next point, and NUMBER\_OF\_POINTS defines the number of points generated. For example:

LOCATION3=MULTILINESTRING;44,1,-93,1;0.1;3

will generate a column called LOCATION2, with three points: e.g., "MULTILINESTRING((-92.456543 44.500763, -92.358408 44.596585, -92.443079 44.592947))".

***POLYGON\****

Used to specify a POLYGON value conforming to the WKT standard.

COLNAME=POLYGON;BASE LATITUDE,DELTA LATITUDE,BASE LONGITUDE,DELTA LONGITUDE;DELTA LENGTH;NUMBER OF POINTS

where POLYGON identifies the data type, BASE LATITUDE is the base latitude value, DELTA\_LATITUDE defines the upper latitude bound, BASE\_LONGITUDE is the base longitude value, DELTA\_LONGITUDE defines the upper longitude bound (these four values define a bounding box with which the first points of each linestring will be generated). DELTA\_LENGTH defines the upper range between the previous point and the next point, and NUMBER\_OF\_POINTS defines the number of points generated (currently, this is restricted to 3). The shape is closed by repeating the first point generated. For example:

LOCATION3=POLYGON;44,1,-93,1;0.1;3

will generate a column called LOCATION3, with three points: e.g., "POLYGON((-92.465912 44.556244, -92.382956 44.578815, -92.422449 44.638159, -92.465912 44.556244))".

\*Note that IOC 16 appears to result in a parsing error unless these are wrapped in double quotes so, currently, the script will add these automatically.

***PERL***

This type can be used to define custom Perl extensions for column definitions not handled by the other types; with this column definition type you are limited only by the Perl language itself. For context you can access the current value of other columns with 'get\_current\_value(col\_number)'. The value of any valid properties of other columns can be retrieved using ‘get\_property\_value(col\_number,property\_name)’. You can also access global properties such as the current row count, the current iteration, etc. (e.g., $g\_var->{‘row\_count’}).

COLNAME=PERL;function\_name;arglist(comma-separated)

where PERL identifies the data type, 'function\_name' is the name of a function that you write in 'generic\_csv\_writer.pl' and ‘arglist’ is a comma-separated list of arguments to be passed to ‘function\_name’.  There is a 'CUSTOM PERL EXTENSIONS' section where you define your custom functions. For example:

PERMITNAME=PERL;get\_permit\_name;1,9

defines a function called 'get\_permit\_name' with two (optional) arguments: 1 and 9. The following definition of 'get\_permit\_name' will combine the current values of column 1 and column 9:

sub get\_permit\_name  
{  
 my $col\_details = shift;  
 my @args = @{$col\_details->{'args'}};  
 my ($first\_col,$second\_col) = (1,2);

$first\_col = $args[0] if defined($args[0]);  
 $second\_col = $args[1] if defined($args[1]);  
 # set value to current value of column x and current value of column y, save in 'current\_value' property  
 $col\_details->{'current\_value'} = get\_current\_value($second\_col) . "\_" . get\_current\_value($first\_col);  
 return $col\_details->{'current\_value'};  
}  
  
This example returns the current time minus 21 hours:

ENDDATETIME=PERL;currentTime\_minus\_x;21,hours  
  
sub currentTime\_minus\_x  
{  
    my $col\_details = shift;  
    my ($count,$units) = @{$col\_details->{'args'}};  
    my $now = DateTime->now;  
      
    $now->subtract($units=>$count);  
    $col\_details->{'current\_value'} = $now->strftime("%Y-%m-%d %H:%M:%S");  
    return $col\_details->{'current\_value'};  
}  
  
The following is a slightly more complicated example that defines a POINT which tracks left to right from a base point, staying within a bounding box. It takes six (optional) arguments:

EVENTLOC=PERL;set\_event\_loc;44,0.01,-93,0.01,0.1,0.1

# Initialize values  
our $event\_loc = {  
    'base\_lat' => 44,  
    'delta\_lat' => 0.01,  
    'base\_lon' => -93,  
    'delta\_lon' => 0.01,  
    'max\_delta\_lat' => 0.1,  
    'max\_delta\_lon' => 0.1,  
    'curr\_lat' => 44,  
    'curr\_lon' => -93  
};  
  
sub set\_event\_loc  
{  
 my $col\_details = shift;  
 my @args = @{$col\_details->{'args'}};

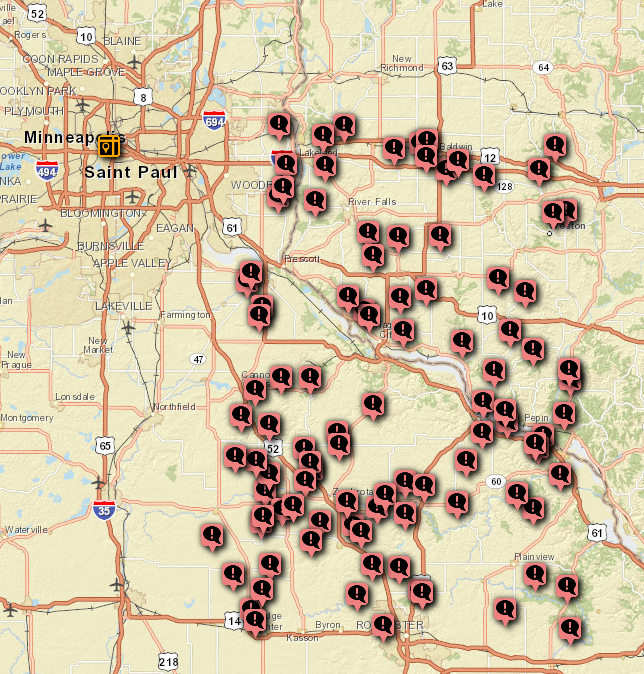
if ($g\_var->{'row\_count'} == 1) { # Initialise if on the first record  
 $event\_loc->{'base\_lat'} = $args[0] if defined($args[0]);  
 $event\_loc->{'delta\_lat'} = $args[1] if defined($args[1]);  
 $event\_loc->{'base\_lon'} = $args[2] if defined($args[2]);  
 $event\_loc->{'delta\_lon'} = $args[3] if defined($args[3]);  
 $event\_loc->{'max\_delta\_lat'} = $args[4] if defined($args[4]);  
 $event\_loc->{'max\_delta\_lon'} = $args[5] if defined($args[5]);  
 $event\_loc->{'curr\_lat'} = $event\_loc->{'base\_lat'};  
 $event\_loc->{'curr\_lon'} = $event\_loc->{'base\_lon'};  
 }

$event\_loc->{'curr\_lon'} += $event\_loc->{'delta\_lon'} unless $g\_var->{'row\_count'} == 1;  
 if ($event\_loc->{'curr\_lon'} - $event\_loc->{'base\_lon'} > $event\_loc->{'max\_delta\_lon'}) {  
 $event\_loc->{'curr\_lon'} = $event\_loc->{'base\_lon'};  
 $event\_loc->{'curr\_lat'} += $event\_loc->{'delta\_lat'};  
 }  
 $col\_details->{'current\_value'} = sprintf("POINT(%.6f %.6f)",$event\_loc->{'curr\_lon'},$event\_loc->{'curr\_lat'});  
 return $col\_details->{'current\_value'};  
}

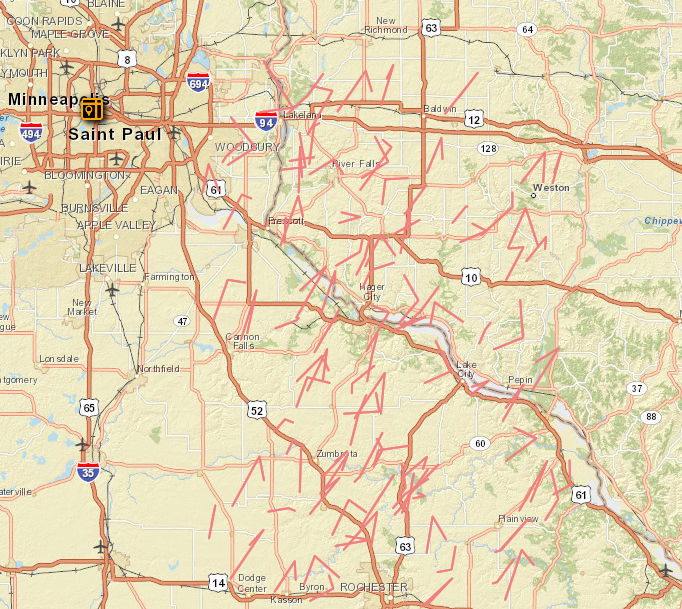
There is no limit to what you can define here (for example, loading information from source files).

**IOC16 Examples**

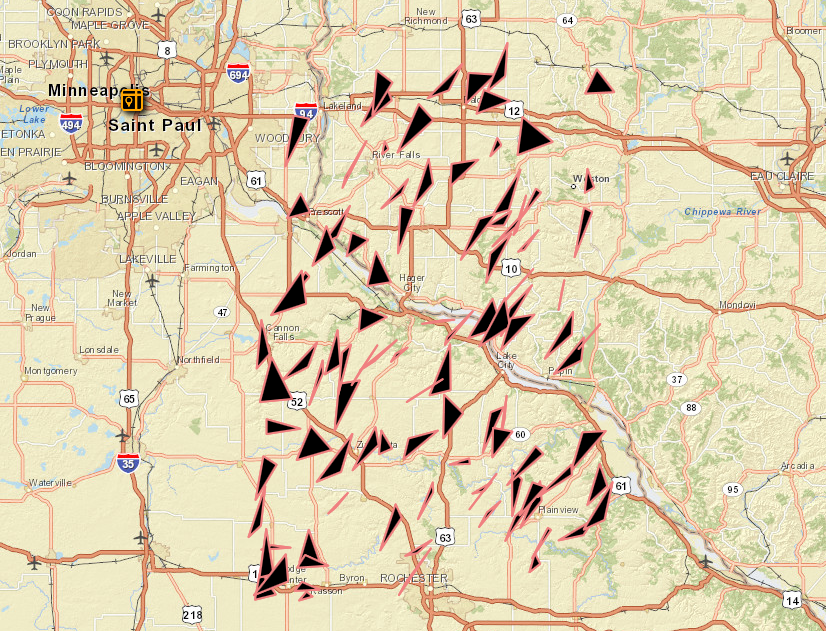
100 POINT values generated according to "LOCATION=POINT;44;1;-93;1”:



100 LINESTRING values generated according to "LOCATION2=LINESTRING;44,1,-93,1;0.1;5".



100 POLYGON shapes generated according to: "LOCATION3=POLYGON;44,1,-93,1;0.1;3".



**Sample Configuration File**

[OPTIONS]  
  
filename=ds\_input\_%d.csv  
dest\_folder=/opt/IBM/ioc/csv  
csv\_delimiter=,  
iterations=1  
records\_per\_file=100  
secs\_between\_iterations=10  
datetime\_format=%Y-%m-%d %H:%M:%S.000  
date\_format=%Y-%m-%d  
time\_format=%H:%M:%S.000  
quote\_string\_values=0  
enum\_separator=,  
  
[COLUMNS]

INDEXNUM=SEQUENCE;1;1;0;0  
STARTDATETIME=PERL;currentTime\_minus\_x;22,hours  
ASSET=STRSEQ;LOC%s;1;1;6;10  
ENDDATETIME=PERL;currentTime\_minus\_x;21,hours  
LOCATION=POINT;44;1;-93;1  
NAME=ENUM;Hurricane Approaching,Heavy Weather,Large Wildfire Moving Rapidly West,Chemical Spill Near Residential Area,Major Water Leak,Multi-vehicle Accident with Injuries  
LASTUPDATEDATETIME=PERL;now  
TIMEZONEOFFSET=0  
PERMITID=INTEGER;1000;10000  
PERMITNAME=PERL;get\_permit\_name;1,9  
READING=FLOAT;0;50

## Sample Output

INDEXNUM,STARTDATETIME,ASSET,ENDDATETIME,LOCATION,NAME,LASTUPDATEDATETIME,TIMEZONEOFFSET,PERMITID,PERMITNAME,READING  
1,2013-11-02 17:29:48,LOC000001,2013-11-02 18:29:48,POINT(-92.695557 44.492279),Large Wildfire Moving Rapidly West,2013-11-03 15:29:48,0,8722,8722\_1,3.302002  
2,2013-11-02 17:29:48,LOC000002,2013-11-02 18:29:48,POINT(-92.365936 44.745972),Heavy Weather,2013-11-03 15:29:48,0,5723,5723\_2,20.768738  
3,2013-11-02 17:29:48,LOC000003,2013-11-02 18:29:48,POINT(-92.238159 44.850342),Heavy Weather,2013-11-03 15:29:48,0,8703,8703\_3,39.163208  
4,2013-11-02 17:29:48,LOC000004,2013-11-02 18:29:48,POINT(-92.528992 44.396729),Major Water Leak,2013-11-03 15:29:48,0,6064,6064\_4,4.676819  
5,2013-11-02 17:29:48,LOC000005,2013-11-02 18:29:48,POINT(-92.972382 44.339111),Large Wildfire Moving Rapidly West,2013-11-03 15:29:48,0,6156,6156\_5,21.200562  
6,2013-11-02 17:29:48,LOC000006,2013-11-02 18:29:48,POINT(-92.493469 44.541809),Hurricane Approaching,2013-11-03 15:29:48,0,5714,5714\_6,49.896240  
7,2013-11-02 17:29:48,LOC000007,2013-11-02 18:29:48,POINT(-92.163971 44.763947),Heavy Weather,2013-11-03 15:29:48,0,7861,7861\_7,17.666626  
8,2013-11-02 17:29:48,LOC000008,2013-11-02 18:29:48,POINT(-92.531281 44.522797),Multi-vehicle Accident with Injuries,2013-11-03 15:29:48,0,2777,2777\_8,18.673706  
9,2013-11-02 17:29:48,LOC000009,2013-11-02 18:29:48,POINT(-92.131195 44.612671),Multi-vehicle Accident with Injuries,2013-11-03 15:29:48,0,1983,1983\_9,39.373779  
10,2013-11-02 17:29:48,LOC000010,2013-11-02 18:29:48,POINT(-92.142731 44.692413),Hurricane Approaching,2013-11-03 15:29:48,0,3799,3799\_10,20.603943  
11,2013-11-02 17:29:48,LOC000001,2013-11-02 18:29:48,POINT(-92.484619 44.751190),Heavy Weather,2013-11-03 15:29:48,0,3955,3955\_11,30.485535  
12,2013-11-02 17:29:48,LOC000002,2013-11-02 18:29:48,POINT(-92.675598 44.627441),Large Wildfire Moving Rapidly West,2013-11-03 15:29:48,0,3787,3787\_12,19.200134  
13,2013-11-02 17:29:48,LOC000003,2013-11-02 18:29:48,POINT(-92.436951 44.611664),Large Wildfire Moving Rapidly West,2013-11-03 15:29:48,0,9520,9520\_13,8.544922  
14,2013-11-02 17:29:48,LOC000004,2013-11-02 18:29:48,POINT(-92.518494 44.293488),Multi-vehicle Accident with Injuries,2013-11-03 15:29:48,0,1262,1262\_14,25.053406  
15,2013-11-02 17:29:48,LOC000005,2013-11-02 18:29:48,POINT(-92.215088 44.966339),Chemical Spill Near Residential Area,2013-11-03 15:29:48,0,9807,9807\_15,34.413147  
16,2013-11-02 17:29:48,LOC000006,2013-11-02 18:29:48,POINT(-92.847504 44.571136),Large Wildfire Moving Rapidly West,2013-11-03 15:29:48,0,9586,9586\_16,33.729553  
17,2013-11-02 17:29:48,LOC000007,2013-11-02 18:29:48,POINT(-92.603851 44.047699),Heavy Weather,2013-11-03 15:29:48,0,7528,7528\_17,40.168762  
18,2013-11-02 17:29:48,LOC000008,2013-11-02 18:29:48,POINT(-92.378082 44.710999),Heavy Weather,2013-11-03 15:29:48,0,5393,5393\_18,7.327271  
19,2013-11-02 17:29:48,LOC000009,2013-11-02 18:29:48,POINT(-92.322174 44.331390),Heavy Weather,2013-11-03 15:29:48,0,9098,9098\_19,47.393799  
20,2013-11-02 17:29:48,LOC000010,2013-11-02 18:29:48,POINT(-92.784882 44.021881),Major Water Leak,2013-11-03 15:29:48,0,8014,8014\_20,30.679321