Supreme Court of Florida

No. AOSC14-17

IN RE: JUROR SELECTION PLAN: ALACHUA COUNTY

ADMINISTRATIVE ORDER

Section 40.225, Florida Statutes, provides for the selection of jurors to serve within the county by an "automated electronic system." Pursuant to that section, the chief judge of the circuit must review and consent to the juror selection process, and the clerk of the circuit court must submit to the Supreme Court of Florida a description of the method for selecting jurors. Section 40.225(3), Florida Statutes, charges the Chief Justice of the Supreme Court with the review and approval of the proposed juror selection process, hereinafter referred to as the "juror selection plan."

The use of technology in the selection of jurors has been customary within Florida for more than 20 years and the Supreme Court has developed standards necessary to ensure that juror selection plans satisfy statutory, methodological, and due process requirements. The Court has tasked the Office of the State Courts

Administrator with evaluating proposed plans to determine their compliance with those standards.

On February 28, 2014, the Clerk of the Circuit Court for Alachua County submitted the Alachua County Juror Selection Plan for review and approval in accordance with section 40.225(2), Florida Statutes. The proposed plan reflects changes to both hardware and software used for juror pool selection in Alachua County.

The Office of the State Courts Administrator has completed an extensive review of the proposed Alachua County Juror Selection Plan, including an evaluation of statutory, due process, statistical, and mathematical elements associated with selection of jury candidates. The plan meets established requirements for approval.

Accordingly, the attached Alachua County Juror Selection Plan, submitted on February 28, 2014, by The Honorable J. K. Irby, Clerk of the Circuit Court, Alachua County, and approved by The Honorable Robert E. Roundtree, Chief Judge of the Eighth Judicial Circuit, is hereby approved for use.

DONE AND ORDERED at Tallahassee, Florida, on March 11, 2014.

Ricky Polston, Chief Justice

ATTEST:

John A. Tomasino, Clerk of Court

CLERK OF THE CIRCUIT AND COUNTY COURTS

EIGHTH JUDICIAL CIRCUIT
ALACHUA COUNTY COURTHOUSE
201 EAST UNIVERSITY AVENUE
GAINESVILLE, FLORIDA 32601

J.K. IRBY
CLERK

TELEPHONE:
352-374-3636

February 28, 2014

Court Services
Office of the State Courts Administrator
500 South Duval Street
Tallahassee, Florida 32399-1925

Re: Juror Selection Application

Dear Court Services:

Enclosed for your review is the Juror Pool Selection Plan for Alachua County. Included is the proposed hardware, software, random number generator program and algorithms to be used in the jury selection process. We are seeking Office of the State Courts Administrator review and Supreme Court approval of this process.

The Honorable Robert E. Roundtree, Chief Judge of the Eighth Judicial Circuit, has reviewed the attached documentation. A copy of his letter indicating his approval is attached.

Please let me know if you need any additional information. I look forward to the Court's approval of this proposed jury pool selection plan.

Respectfully,

J. K. "Buddy" Irby
Alachua County Clerk of Court

xc: The Honorable Robert E. Roundtree, Chief Judge , Eighth Judicial Circuit
Ted McFetridge, Court Administrator, Eight Judicial Circuit

Eighth Judicial Circuit of Florida

Alachua, Baker, Bradford, Gilchrist, Levy and Union Counties

Chambers of
Robert. E. Roundtree, Jr.
Chief Judge

Alachua County Courthouse
Family and Civil Justice Center
201 East University Avenue
Gainesville, Florida 32601
(352) 374-3644
(352) 374-3640 (fax)

Karen A. Wable
Judicial Assistant

February 26, 2014

Court Services
Office of the State Courts Administrator
Supreme Court Building
500 South Duval Street
Tallahassee, FL 32399

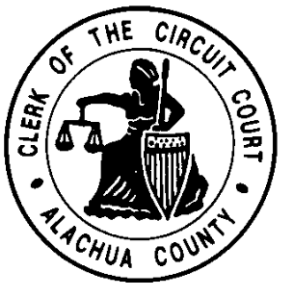
Dear Court Services:

By this letter, I am certifying that I have reviewed and consent to the proposed jury pool selection plan which is attached.

Sincerely,

Robert E. Roundtree, Jr.

Enclosure



Alachua County

Jury Selection Plan

February, 2014

Table of Contents

Purpose and Scope

Purpose and Scope

The purpose of this document is to describe the design and implementation of the jury selection process to be used in a new jury management system (JuryView) for Alachua County, Florida.

The scope of this document includes the algorithms and methods used to:

- Create and maintain a master candidate table, and
- Select a set of names from the master candidate table to create a jury pool.

Also included in the scope is a description of the equipment, operating system and programming software, methods, and modes of operation to be used in the jury selection process.

Sufficient detail will be provided to satisfy the statutory condition of selection "by lot and at random" and of due process as required by chapter 40, Florida Statutes.

Initial Candidate Selection List Creation List

A. Source Lists

The initial candidate selection list is created from:

1. The quarterly list of names provided by the Department of Highway Safety and Motor Vehicles (DHSMV) pursuant to section 40.011, Florida Statutes. DHSMV provides the Florida Court Clerks and Comptrollers (FCCC) with a data file that contains all persons holding a valid Florida Driver License or Identification Card. FCCC acts as a distribution agent, grouping the data by County and dispersing it to the Clerk of Court. This data file only includes Alachua County residents.
2. The existing juror candidates from the existing juror management system along with qualification data, which will be imported from Alachua County's current jury selection system. The DHSMV data file will be processed against the existing, adding new names and updating existing information, such as addresses.
3. Manually entered candidates from the existing juror candidate data. JuryView allows individuals to be manually entered as required.

In addition:

4. The Clerk is the official custodian for the Master Candidate table (source list) and is responsible for the security and integrity of the list and all processes.
5. The Master Candidate Table (final juror candidate list) will not contain less than 250 names per section 40.02, Florida Statutes.

B. Source Data Adjustments

1. The clerk's office receives the DHSMV source data from FCCC quarterly via a dedicated network connection and the file is placed into a secured folder on a server for processing. The data received is an ASCII, fixed-length file and is not modified in any way.
2. When names are imported, they are matched with existing data and the contact information is updated in JuryView, but the name retains JuryView's exclusion codes.
3. FCCC also provides a data file from the Office of Vital Statistics to the clerk twice a month that contains a list of persons who are deceased. This is also an ASCII, fixed-length file and is also placed into a secure folder on a server for processing. It is not modified in any way. Once

received, the data is imported and parsed and used to update the candidates table in JuryView. Each deceased person is flagged as permanently excused and no longer eligible for selection. These flagged names are kept and compared with each quarterly load.

4. The clerk's jury manager receives a monthly report of persons who are determined incapacitated or incompetent from the Clerk's probate department. These names are manually flagged as permanently excused and kept and compared with each data load.
5. FDLE provides felony convictions to the clerk monthly, and the names on the felony conviction list are manually flagged in JuryView as permanently excused and kept and compared with each data load.
6. The DHSMV file gives age information and persons can request removal from the candidate list if they are age 70 or older. These exclusions are entered manually, flagged as permanently excused, and kept and compared with each load.
7. For those people who have filed request-to-serve affidavits, their information is manually entered into the candidates table by the clerk's office and the names are flagged so that they are not removed in later quarterly updates.
8. Candidate names are marked with exclusion codes based on the exclusion lists identified above per sections 40.013, 40.022 or 40.023, Florida Statutes. Candidate data also retains service information, such as whether a candidate is on an open panel. Candidate names are flagged as excluded if they are prohibited from serving, have already served, or their service is otherwise excused or postponed. JuryView also allows exemption entries, such as those over 70 or caretakers, and allows for postponements so that those candidates will be included in later venires. For example, if an individual is postponed until January 15, 2014, due to a scheduling conflict, that postponement date is stored in their record and the first venire created on or after that date will automatically include that individual in addition to those that were selected at random. Once the candidate is in a pool, they cannot be pulled into a new term automatically, but can be manually added to the retainer pool (excused jurors with future availability dates) and when the 'retainer pool' box is checked, these jurors are selected first for the next pool of candidates.

C. Time Frame

1. The DHSMV file is provided quarterly: January, April, July, and October. The master Candidate table is updated with this information when received.
2. The file containing the names of deceased persons is provided on the 1st and the 15th of each month. The Master Candidate table is updated with this information when it is received.
3. Reports on incompetent persons and convicted felons are received monthly.
4. Those filing affidavits asking to serve as jurors are entered manually by the clerk of court as the affidavits are received.
5. Candidates that are excused pursuant to section 40.013, Florida Statues, or postponed to a later date are updated on demand.

D. Initial Jury Candidate List

1. JuryView's initial database creation process loads DHSMV data from the work table into the permanent tables [Impjuror] in the application using the SQL Server script to transfer the single table structure to an SQL server 2008 R2 64-bit relational database. JuryView identifies duplicates at the point of loading the DHSMV file. Duplicates are extracted and placed in a

separate table [Jurordup]. JuryView can be configured to sort out zip codes that are not Alachua County 326 zip codes to find errors in the data sent by FCCC, which is already sorted by county. Once this juror database is built the first time, all quarterly updates will be performed against it. This initial creation will be scheduled to coincide with the receipt of quarterly DHSMV data. Each name is assigned a unique id number as it is imported in or added to JuryView.

2. This application is being acquired to replace an existing application. Once the initial database is created, specified data will be imported from the current system. Imported data includes (a) records of candidates that are not qualified to serve, such as felons, the deceased, and the incapacitated; (b) those who have served within the last year, (c) those over 70 who have requested exclusion; (d) those who have been added by affidavit; and (e) those whose upcoming service is scheduled. No names will be modified during this import.
3. With quarterly updates, each new name found in the DHSMV data, is added to the database and assigned a unique id number. If the name exists in the database, the person's information is updated with the DHSMV data. Jurors that exist in the database, but are not in the quarterly DHSMV list are blocked from service, unless they were manually added as part of the affidavit process. [When a juror is in the database but not in the incoming file, they get a temporary excusal of IMPEXJUR for the following quarter—the date range comes from the quarterly refresh date.] The quarterly data refresh does not override excusal or deferral flags.
4. Those filing affidavits asking to serve as jurors are entered manually by the Clerk of Court as the affidavits are received.
5. Exclusions:
    a. The Department of Health file with names of the deceased is provided by FCCC. This file is reviewed and the listed names are manually flagged as excluded.
    b. The other required exclusions are also manually flagged upon receipt of the incapacity and convicted felon reports.
    c. Deferrals are also flagged with a date so that when a deferred juror is reviewed for selection, the name is excluded if the time on the deferral has not expired.
    d. Jurors who have served are also flagged with a 12-month deferral and will be passed over for selection until the deferral date has expired.

Name Selection

A. Equipment and Software

JuryView is a graphical, client-server-based application written in Uniface. The current version is 2.51 and is owned by CourtView. It is a commercial application, which is also being used in Sarasota and Flagler Counties.

The application will be housed in a virtual server hosted within a Microsoft Hyper-V Cluster on Dell PowerEdge 710 servers. The database server is also hosted in a clustered Dell PowerEdge 710 servers running SQL Server 2008 R2 (64 bit). Both clusters are running Microsoft Windows Server 2008 R2 (64 bit).

Changes in the clerk's operating system will not affect the software.

B. Security

The Clerk of Court is the official custodian and is responsible for the security and integrity of the name lists and of the juror pool selection process.

All servers are located in the Clerk's Office located in the Alachua County Civil and Family Justice Center. The Clerk's Office has a designated Data Center which is secured by a code-key lock. Only authorized personnel have access to the Clerk's Data Center. The Data Center is designed for 24/7 operation and is supported by an uninterruptible power supply and generator systems.

JuryView is a client-server-based application and, as such, is not accessible to the public. The software application is secured via passwords, set to allow only the appropriate clerks access to the application. Court Administration will be supplied their own passwords to the application so they can continue to process excusals and postponements as is done under the current application.

All information necessary for auditing purposes, including random number seed sets is stored perpetually in tables as part of the application base.

C. Process Overview for Name Selection

The clerk's office and a judge determine that a jury pool needs to be created and how many potential jurors the pool should contain. Information about this pool is entered into JuryView such as the date, number of jurors needed, etc. JuryView can select whatever number of jurors is needed as often as needed.

JuryView begins the selection process by assigning a random number to each juror record with a status code of 'qual.' Uniface then sorts the jurors by the randomly assigned numbers in sequential order. Then the program filters out any juror that is not eligible to serve, which is based on the juror's status. Individuals are then selected based on their number until the predetermined or requested population is filled. This deferral process is repeated for every term and panel selection. This is a more detailed overview:

1. The RNG is initialized with the seed set (see Section E Initialization).
2. A request is made to the RNG for a random number.
3. The random number is assigned to the first record in the candidate table with a status of 'qualified.' Should the RNG return a number that has already been selected, the process returns to step 1.
4. If there are unassigned records with a status of 'qualified,' the process returns to step 1, otherwise it ends.
5. A record set [Master Juror Table] is created with an SQL Select against the master Candidate table. The SQL Select contains the necessary select/omit logic so as to filter out any juror that has an excusal with begin and end dates that do NOT fall in the panel date range, including a person who has served within the past year to serve in this jury pool. [The definition of excused is a juror that has an EXEMPT_CD not null and EXEMPT_START_DT and EXEMPT_END_DT are populated in the JUROR table.] This deferral process is repeated for every term/panel selection.
6. JuryView then assigns individuals to jury pools from the Master Juror table created in section 5, with jurors selected in juror number order based on the random number that was assigned, the lowest number being selected first, until each pool has the needed number of jurors.

D. Name Selection Algorithm

Random Number Generator

The Random Number Generator (RNG) used in JuryView is the Mersenne Twister (MT)[1]. The routine was re-written in C++ to produce the desired results within the JuryView application and a copy of the source code is included (Attachment A). The jryrndgn routine (attached) receives five parameters. One parameter is the term, service or panel ID. The other four parameters are the seed values generated by the MicroSoft SQL Server Rand function. Each seed value is generated by repeated invocation of the following TSQL statement:

```
SELECT @seed1=CONVERT(int, RAND() * 1000) * 1000000
        + CONVERT(int, RAND() * 1000) * 10000
        + CONVERT(int, RAND() * 1000) * 1
        + (DATEPART(ms, GETDATE()) * 10000) * 100
```

See Appendix B, Code for the seedgen procedure along with notes from the original tpr that created this process.

Certification

The output of the version of the Mersenne Twister used in the CourtView application JuryView 2.51 was verified against the test output provided by the authors in the file: mt19937ar.out available on the authors web site: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

The generator was initialized with the values <0x123, 0x234, 0x345, 0x456> and 1000 numbers were generated from the function genrand_int32. The last fifteen values are provided for comparison.

3759461013	981457956	830587662	877191791	650996736
988064871	515461600	089077232	225147448	249609188
2643151863	896204135	416995901	397735321	460025646

E. Initialization

RNG

The RNG is written in C++ and all other processes are written in MS SQL Server SQL language and it is compiled as a stored procedure in the Master Jury database. This process is explained in the following example:

A list containing all potential juror names in Alachua County is loaded into JuryView from the acceptable source data (DHSMV). At the time the venires are ready to be built JuryView assigns a random number to each qualified individual until the requested number of jurors is filled. These jurors will be "skipped" at the next venire list creation because they have already been selected.

Parameters

The other four parameters are the seed values generated in the stored procedure seedgen which generates four seeds by using the MicroSoft SQL Server RAND function.[2] The formula builds each

[1] Developed by Makato Matsumoto and Takuji Nishimura in 1996, with improved initialization in 2002. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

[2] SQL Server 2008 R2 version explained at http://msdn.microsoft.com/en-us/library/ms177610(v=sql.105).aspx

seed by using the RAND function to generate a number between 1 and 999 that is then multiplied by 1000000 then takes that number and adds it to a new RAND number between 1 and 999 which is multiplied by 1; this number is then added to a number generated from the millisecond of the current time multiplied to 10000 then multiplied by 100. This formula is used to generate each seed value.

The random key for each juror is generated by the Mersenne Twister using jryrngdn.exe and is populated using the RAND_NBR.RANDNBR field. The random key value is only updated for each juror that meets the cert level (term, service, and panel) requirements that the juror is qualified for. The following scripts: rand_term_juror, rand_service_juror, and rand_panel_juror are used to populate the random key field on each juror. Each script has detailed descriptions of how these values are generated. The quarterly draw generates the random key value on the JURY_POOL table with the same formula from the seedgen procedure that generates the seed values.

The four seed values (seed1, seed2, seed3, seed4) in the RANDSEED table are 32-bit values. Please see addendum.

Random numbers (rand_nbr) are created for jurors that meet certain criteria (status_cd = 'QUAL'), which you can see in the RAND_TERM_JUROR procedure, because these are the only jurors that can be selected in the uniface code. The RANDJURY (Attachment C) and RAND_TERM_JUROR (Attachment D) procedures and the seedgen procedure (Attachment B) referenced above are the 3 SQL procedures used in the randomization.

Starting, Stopping and Other Factors

The process described for the Name Selection Algorithm has no start or stop features. Each name is assigned a random number. In the event of a failure in the system during this processing, the venire list can be recreated on demand. A record of the old and new seed values for the RNG are stored for auditing purposes.

If a failure happens during the creation of the venire all random number are assigned first before any juror is place on a venire so the randomization will have already been completed. But if the list is only partially completed there is no proper method to fill it manually and keep the true randomization so the best process for any venire that did not complete would be to delete the unfinished venire and re-run the process which will again regenerate random numbers and assign those to qualified jurors until the process properly completes for the full amount of requested jurors.

APPENDIX A

Random Number Generator Algorithm for Initial Seed Value

```
#include <stdio.h>
#include <fglsys.h>
#include <ifxtypes.h>

#define  TAB_SIZE 97
#define  NO_OF_BITS 24
#define  MOD_1 179
#define  MOD_2 169
#define  CONST_1 53
#define  CONST_2 362436
#define  CONST_3 7654321
#define  CONST_4 16777213
#define  CONST_5 96
#define  CONST_6 32
#define  CONST_7 16777216.0
#define  MASK 0xffffff
#define  RAND_TABLE "/usr/tmp/rtable"

struct table
{
      long c_val;
      short i_ind;
      short j_ind;
      float uni_tabl[TAB_SIZE];
};

/* This function creates the 97 row table of floating values which is used
   later in the UNI function.  The table UNI_TABL is saved in a non-printable
   file along with 3 array counters, C_VAL, I_IND, and J_IND.  This file will
   be read in again in the UNI function each time that the calling program
   needs a new random number.  The table, I_IND, and J_IND are all modified
   in the UNI function and saved to the file upon exit so that the changed
   contents of the file can be read back in when the next random number is
   requested by the calling program.                                    */

int set_uni(params)
   int params;
{
      short             i, j, k, l;
      struct table      table_str;
      register int      i1, j1;
      long        rem, rem1;
      unsigned long     sum;
      FILE        *t_ptr;
      short             status = 0;

      /* This file is used to store the table.  It is read back into
         the UNI function later.                                    */

      if ((t_ptr = fopen(RAND_TABLE, "w+")) == NULL)
      {
            /* fprintf(stderr, "Can't create rtable for writing\n");*/
        status = -1;
      }
```

```
      else  /* rtable open for writing */
      {
        if (params == 4)
        {

          popshort(&l);       /* pop seed numbers off the stack */
          popshort(&k);
          popshort(&j);
          popshort(&i);

          /*************************************************************/
          /* This section is from the original C program on the Targon. */
          /* The logic has not been modified at all.                 */

          for (i1 = 0; i1 < TAB_SIZE ; i1++)
          {
            sum = 0;
            for (j1 = NO_OF_BITS; j1 > 0; j1--) {
                  rem1 = (long)((i * j) % MOD_1);
                  rem1 = (long)((rem1 * k) % MOD_1);
                  i = j;
                  j = k;
                  k = (short)rem1;
                  rem = (long)(((CONST_1 * l) + 1) % MOD_2);
                  l = rem;
                  sum *= 2;
                  rem = (long)((l*rem1) % 64);
                  if (rem >= CONST_6)
                        sum += 1;
            }
            table_str.uni_tabl[i1] = sum;
          }
          table_str.c_val = CONST_2;
          table_str.i_ind = CONST_5;
          table_str.j_ind = CONST_6;
          /*                                                  */
          /*************************************************************/

          /* Now save the table to a file so it can be read in later.  */

          fwrite (&table_str, sizeof(table_str),1,t_ptr);
          fclose (t_ptr);
        }
        else            /* 4 parameters were not passed to function     */
        {
          status = -2;
        }
      }
  retshort(status);     /* Push the status onto the stack.  This lets the */
  return(1);            /* calling 4gl know if the C function executed.   */
}

/* This function reads in table from the file, gets the next random number,
   and then saves the new modified record.                          */

int uni(int params)
{
```

```
    struct table      table_str;
    long        uni_val;
    FILE        *t_ptr;
    double            uni_num;
    long        voter_number;
    long        stemp = 0;
    short       status = 0;

if ((t_ptr = fopen(RAND_TABLE, "r+")) == NULL)


{
    /* fprintf(stderr, "Can't open randtable for read update\n");*/
  status = -3;
}
else      /* File opened for read */
{
  if (params == 1)    /* 1 parameter passed to function */
  {
    poplong(&voter_number); /* Pop voter reg number off of stack */
    fread (&table_str, sizeof(table_str),1,t_ptr); /* Read table */

    /*************************************************************/
    /* This section is from the original C program on the Targon */
    /* The logic has not been modified at all.                   */

    uni_val = table_str.uni_tabl[table_str.i_ind] -
      table_str.uni_tabl[table_str.j_ind];
    uni_val &= MASK;
    table_str.uni_tabl[table_str.i_ind] = uni_val;

    if (table_str.i_ind == 0)
    table_str.i_ind = CONST_5;
    else
    table_str.i_ind--;

    if (table_str.j_ind ==0)
      table_str.j_ind = CONST_5;
    else
      table_str.j_ind--;

    table_str.c_val -= CONST_3;
    if (table_str.c_val < 0)
    table_str.c_val += CONST_4;
    uni_val -= table_str.c_val;
    uni_val &= MASK;

  /*************************************************************/

  /* Now rewind the file, save the table, close the file, and
     return to the calling program.                           */

    rewind(t_ptr);
    fwrite (&table_str, sizeof(table_str),1,t_ptr);
    fclose (t_ptr);

          /* Calculate random number */
    stemp = (long) ((uni_val / CONST_7) * voter_number) + 1;
```

```
    }
    else          /* 1 parameter was not passed to function */
    {
      status = -4;
    }
  }
  retlong(stemp); /* Push random number onto stack and*/
  retshort(status);     /* push status onto stack for 4gl */
  return(2);
}
```

Appendix B

The code for the seedgen procedure along with notes from the original tpr that created this process below:

```
CREATE PROCEDURE seedgen @nbr int, @group char(10),  @group_id int, @SeedId int OUT
AS
SET NOCOUNT ON
SET ROWCOUNT 0
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--seedgen.ms2
--
--Steve Smith
--April 18, 2002
--
--Generates four random number seeds and inserts them into randseed.
--
--Parameters:
-------------
--@nbr: The number of numbers to generate using the 4 seed values.
--
--@group: The type of group the numbers are being generated for.
--
--@group_id: The id of the group.
--------------------------------------------------------------------------------
--History:
----------
--04 FEB 2003       SJS
--
--Added the commit at the end of the procedure.  For some reason, a hanging
--transaction was being created that locked the row created by this proc.  This
--lock would then prevent the c routine from retrieving the record from the
--table and using the seeds.
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
 -------------------------Working Variables-----------------------------------
DECLARE @seed1 int,     @seed2 int,         @seed3 int,
        @seed4 int,     @randseed_id int,   @ins_by char(12),
        @term_id int,   @service_id int,    @panel_id int
--------------------------------------------------------------------------------
--Get the user name of the person logged into the database.--
---------------------------------------------------------------
SELECT @ins_by = SUSER_SNAME()
--Load the id of the group into its correct variable--
--for insert into randseed.                        --
-----------------------------------------------------
SELECT @term_id = NULL
SELECT @service_id = NULL
SELECT @panel_id = NULL
IF (@group = 'TERM')
BEGIN
   SELECT @term_id = @group_id
END
ELSE
BEGIN
IF (@group = 'SERVICE')
```

```
   BEGIN
      SELECT @service_id = @group_id
   END
   ELSE
   BEGIN
      IF (@group = 'PANEL')
      BEGIN
         SELECT @panel_id = @group_id
      END
   END
END
--Generate the 4 random seed values.--
---------------------------------------
SELECT @seed1 = CONVERT(int, RAND() * 1000) * 1000000
              + CONVERT(int, RAND() * 1000) * 1000
              + CONVERT(int, RAND() * 1000) * 1
              + (DATEPART(ms,GETDATE()) * 10000) * 100


SELECT @seed2 = CONVERT(int, RAND() * 1000) * 1000000
              + CONVERT(int, RAND() * 1000) * 1000
              + CONVERT(int, RAND() * 1000) * 1
              + (DATEPART(ms,GETDATE()) * 10000) * 100

SELECT @seed3 = CONVERT(int, RAND() * 1000) * 1000000
              + CONVERT(int, RAND() * 1000) * 1000
              + CONVERT(int, RAND() * 1000) * 1
              + (DATEPART(ms,GETDATE()) * 10000) * 100

SELECT @seed4 = CONVERT(int, RAND() * 1000) * 1000000
              + CONVERT(int, RAND() * 1000) * 1000
              + CONVERT(int, RAND() * 1000) * 1
              + (DATEPART(ms,GETDATE()) * 10000) * 100
  --Generate a primary key value for the table.--
-----------------------------------------------
EXEC randjury 'randseed', 99999999, @randseed_id OUT
--Insert into randseed table.--
-------------------------------
INSERT INTO randseed
   (seed1,         seed2,
    seed3,         seed4,
    randseed_id,   u_version,
    act_dttm,      ins_by,
    ins_dttm,      ins_prg,
    upd_by,        upd_dttm,
    upd_prg,       nbr,
    term_id,       service_id,
    panel_id)
VALUES
   (@seed1,        @seed2,
    @seed3,        @seed4,
    @randseed_id,  '!',
    GETDATE(),     @ins_by,
    GETDATE(),     'seedgen',
    @ins_by,       GETDATE(),
    'seedgen',     @nbr,
    @term_id,      @service_id,
```

```
    @panel_id)
--Assign the value of the random seed id parameter.--
-----------------------------------------------------
SELECT @SeedId = @randseed_id
--04 FEB 2003       SJS                    --
--                                         --
--Added commit to prevent c routine from being--
--blocked when trying to retrieve the record. --
-------------------------------------------------
COMMIT

-------------------------End of Proc.----------------------------------------
```

RETURN

This is copied from the original issue that created this process(see tpr 38102 for details):

Notes:
-=-=-=-=-
-Execute flrand.mss first
-Execute randjury.mss second
-Execute seedgen.mss third
-Execute remaining SQL scripts
-JRYRNDGEN.EXE should be placed in the root directory of the JuryView application.
-Notice that we are introducing a URR into the JuryView application (jury2_0n.urr). The ASN file for the application will
need to be edited to include this file. For examples, see the QA CRTV ASN file and the DEV ASN file for Jury 2.0.
Create the following System Parameters:
-=-
These system parameters are used by the application to tell which randomization routine
to use (either SQL -the old way- or using the new C routine). If the system parameters are not
found in the system, the application will default to using the old SQL randomization routine. The value of 'SQL' designates that the SQL routine should be used. The value of 'CRAND' designates that the new C routine should be used.

Group: RANDGEN
Level: GLOBAL
Name: RAND_TERM
Value SQL or CRAND

Group: RANDGEN
Level: GLOBAL
Name: RAND_SERVICE
Value SQL or CRAND

Group: RANDGEN
Level: GLOBAL
Name: RAND_PANEL
Value SQL or CRAND

New C Routine
-=-=-=-=-=-=-=-
The new C routine needs four seed values to produce random numbers. These four seed values are stored in the
randseed table. The id of the group (term, service or panel) that the numbers are being generated for is also stored in

the randseed table. The numbers generated by the four seed values are stored in the randnbr table. When each
number is assigned a juror, the number is logged with the jurors juror_id in the randnbr table.

SQL Routine
-=-=-=-=-=-=-
The SQL routine does not use seeds and does not log which juror received which number. It simply uses the native
random number generator for the DBMS being used to assign jurors random numbers.