Supreme Court of Florida

No. AOSC19-37

IN RE: JUROR SELECTION PLAN: CLAY 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 for compliance with those standards.

By letter dated May 31, 2019, the Clerk of the Court for Clay County submitted the Clay County Juror Pool 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 Clay County.

The Office of the State Courts Administrator has completed an extensive review of the proposed Clay 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 Clay Juror Pool Selection Plan, received on June 6, 2019, from The Honorable Tara S. Green, Clerk of the Circuit Court for Clay County, and approved by The Honorable Mark H. Mahon, Chief Judge of the Fourth Circuit, is hereby approved for use.

DONE AND ORDERED at Tallahassee, Florida, on July 30, 2019.

AOSC19-37 7/30/2019

Chief Justice Charles T. Canady

AOSC19-37 7/30/2019

ATTEST:

AOSC19-37 7/30/2019

John A. Tomasino, Clerk of Court

AOSC19-37 7/30/2019

TARA S. GREEN
Clerk of the Circuit Court, Clay County
825 N. Orange Avenue • P.O. Box 698, Green Cove Springs, FL 32043

OFFICE 904.269.6317 • FAX 904.269.6390
info@clayclerk.com • www.clayclerk.com

RECEIVED
JUN 06 2019

May 31, 2019

Office of the State Courts Administrator
Attn: Mr. Jonathan C. Moody, M.S.
Supreme Court Building
500 South Duval Street
Tallahassee, Florida 32399-1925

Re: Request for Approval of Juror Pool Selection Plan

Dear Mr. Moody:

Enclosed for your review is the Juror Pool Selection Plan for Clay County. The attached proposal includes detailed information regarding the hardware, software, random number generator program and algorithms to be used in the jury process. We are seeking the Office of the State Courts Administrator review and Supreme Court Approval for this process.

The Honorable Mark H. Mahon, Chief Judge of the Fourth Judicial Circuit, has reviewed our recommended proposal. A signed statement indicating his review and approval is attached.

Please let me know if you require any additional information. I look forward to the Court's approval of this proposed Jury Pool Selection Plan.

Respectfully submitted,

Tara S. Green

Tara S. Green

Enclosure

cc: The Honorable Mark J. Mahon
Joseph Stelma, Jr., 4th Circuit Trial Court Administrator

CIRCUIT COURT
FOURTH JUDICIAL CIRCUIT OF FLORIDA
CLAY, DUVAL AND NASSAU COUNTIES

MARK H. MAHON
CHIEF JUDGE

DUVAL COUNTY COURTHOUSE
501 W. ADAMS ST. ROOM 7140
JACKSONVILLE, FLORIDA 32202

May 13, 2019

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

Re: Clay County Jury Selection Plan

Dear Sir or Madam:

In accordance with section 40.225(1), Florida Statutes, I have reviewed the Jury Selection Plan for Clay County dated May 9, 2019, and hereby consent to its use within the Fourth Judicial Circuit.

Sincerely yours,

Mark H. Mahon
Chief Judge

MHM:fg

cc: The Honorable Don H. Lester
Tara S. Green, Clerk of Court Clay County
Joseph Stelma, Jr., 4th Circuit Trial Court Administrator

Clay County

Jury Selection Plan

May 9, 2019

Contents

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 (JuryMark) for Clay County, Florida.
The scope of this document includes the algorithms and methods used to:

- Create and maintain a master candidate table
- 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.

Equipment

a. The Jury Management System (JURYMARK) SQL database runs on a VMWare ESX virtual server running Windows 2012 R2. This server is located in a computer room of the Clay County Courthouse, a secured facility under the control of Clay County's Clerk of Court IT Department. This is also the site where jury trials for criminal cases are heard. There are both test and production instances of the JuryMark application and both run under Windows 2012 RT and Microsoft SQL 2016.

b. The primary JuryMark web/application server will be located at the Clay County Courthouse building in the same computer room as the JuryMark database server. This server is a VMWare ESX virtual server running Windows 2012 R2.

c. The main Jury database is replicated to a secondary jury server at the Clay County Courthouse. This secondary server is a VMWare ESX virtual server running Windows 2012 R2. In addition to the replicated JuryMark database, this server will be configured as a secondary web/application server to execute the JuryMark application in the event of a failure.

d. In the event of a network failure, users in the Clay County Courthouse will execute the application from the replicated database residing on the local server.

e. The JuryMark database will be backed up every 2 hours using VEEAM servers stored to offsite disks and the backups are kept for one month before being overwritten.

f. All hardware and software associated with the jury application will be upgraded on an as needed basis.

Alternative Method of Selecting Venire

a. The sources from which names shall be taken are:

    1. A quarterly updated list of Florida Department of Highway Safety and Motor Vehicle (DHSMV) licensed drivers and identification card holders, 18 years of age or older, who are citizens of the United States, and legal residents of Florida residing in Clay County.
    2. Persons filing affidavits pursuant to §40.011.

b. The Clerk of Circuit Court is designated the official custodian of the computer records to be used in jury selection and shall ensure they are not accessible to anyone other than those directly involved in selection of venires, as herein provided. Functions of the Clerk of the Circuit Court may be performed by her deputies. The Clerk shall maintain these sources in accordance with §40.022 and other relevant statutes, if implemented.

c. The Chief Judge or the Chief Judge's designee, shall direct the Clerk of the Circuit Court to select at random, and as often as required, jury pools of no less than 250 qualified prospective jurors from a file of all Clay County licensed drivers, identification card holders and persons filing affidavits pursuant to §40.011 using the method described in this plan.

d. The Clerk of the Circuit Court in Clay County shall cause jury venires to be selected from the jury pool using the method described in this plan, under supervision of a judge of any court of record or the Chief Judge or his designated representative.

Process for Maintaining & Updating Prospective Juror File

a. Each calendar quarter, the Florida Department of Highway Safety and Motor Vehicles (DHSMV) sends an electronic file of licensed drivers and ID card holders to the Florida Association of Court Clerks (FCCC). Additionally, we receive a deceased list twice a month from DHSMV. The FCCC places all the files on our server to be imported.
b. JuryMark application is used to load the new FCCC File and merge and match the data with existing juror records stored in the JuryMark Database. The JuryMark database includes juror personal data, service history and excusal status. Jurors that are temporarily or permanently excused are flagged as ineligible, but not deleted from the database.
c. The records from the FCCC file are compared to the existing juror records in the JuryMark database using driver's license number. Where a match is found, the addresses are compared and, if different, the DHSMV address replaces the JuryMark address. Records that exist in the FCCC file but not the JuryMark database, are added to the JuryMark Database. Records that exist in the JuryMark database but not the FCCC file, are flagged as inactive in the JuryMark database unless they were created as a result of the filing of an affidavit pursuant to §40.011. Records that are flagged as inactive remain in the JuryMark

database indefinitely but are bypassed by the system during the jury pool selection process.

d. JuryMark contains a status field in the Juror table that determines whether the Juror is qualified for service. The clerk can use the JuryMark application to change the status of a given Juror as needed. The Regular Loading of Jurors into the database does not alter this column so the exclusion status is preserved.
e. Maintenance is conducted twice a month, on the 1st and 15th, to update the JuryMark juror records in accordance with F.S. 40.022 (e.g., identifying convicted felons, deceased persons and legally incapacitated persons, and processing them according to statute). New maintenance procedures will be developed and employed to comply with other relevant statutes when implemented, assuming that data and/or processes from external agencies are available.
f. Persons filing affidavits pursuant to §40.011 will be added to the JuryMark database within two working days.

Juror Selection Process

a. The selection of candidates for weekly petit jury pools is done three (3) weeks in advance of the reporting date. For Grand Jury, the selection of candidates is done twice a year, three weeks in advance of the reporting date.

b. Using JuryMark, the Jury Pool Manager/Grand Jury Clerk or their respective designees, enter the jury pool location and the number of jurors required (minimum of 250 per F.S. 40.02) for the serve date specified. No other information is supplied by the user. JuryMark will select and summons the number of jurors requested and data associated with the selection of a juror pool (e.g., date, number of jurors requested) is stored for future retrieval and reporting.

c. Prior to invoking the process for randomly selecting jurors, JuryMark determines the number of jurors previously postponed, deferred or re-summoned to the serve date specified and subtracts this number from the total number requested. The result is the number of jurors that JURYMARK must randomly select from the juror database. This step in the selection process does not apply to Grand Jury selection.

Step by Step Summary of the JuryMark Random Selection Process

Jurors are selected onto Pools and then from those Pools again onto Panels. Both the Pool and Panel tables contain columns to store the random seed values used to initialize the PRNG to select the Jurors.

The PRNG we are using is the Mersenne Twister algorithm by Makoto Matsumoto and Takuji Nishimura found in the .Net Assembly that can be downloaded from Random OPs here: http://www.hvass- labs.org/projects/randomops/cs/

Validation of the c# implementation of the Mersenne Twister

The output of the version of the Mersenne Twister used in the JuryMark application JuryMark 1.0.0.19 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 first fifteen values are provided for comparison.

1067595299 955945823 477289528 4107218783 4228976476
3344332714 3355579695 227628506 810200273 2591290167
2560260675 3242736208 646746669 1479517882 4245472273

Seeding the Mersenne Twister

JuryMark generates 624 random 32bit seeds for the Mersenne Twister algorithm by using the Microsoft Enhanced Cryptographic Service Provider (CSP) which implements a cryptographically strong random number generator that uses many unpredictable inputs to create the random seeds. By providing many unpredictable inputs to generate the random seeds, the algorithm is able to generate an unpredictable, random starting point for the Mersenne Twister algorithm. Microsoft's System Security Cryptography RNG Crypto Service Provider is the most cryptographically strong random number generator available on the Windows platform. In Microsoft's CSPs, the CryptGenRandom function uses the same random number generator used by other security components. This allows numerous processes to contribute to a system-wide seed.

CryptoAPI stores an intermediate random seed with every user. To form the seed for the random number generator, a calling application supplies bits it might have-for instance, mouse or keyboard timing input-that are then combined with both the stored seed and various system data and user data such as the process ID and thread ID, the system clock, the system time, the system counter, memory status, free disk clusters, and the hashed user environment block.

Note: by default JuryMark will generate 624 seeds, however, this value can be increased by modifying the setting "RandomSeeds" in the Setting Table in the JuryMark Database.

Each Juror in the system is assigned a Juror ID - a unique sequential number in the system using SQL

Server identity column, so the first juror added is 1, the second is 2, and so on. Jurors are never deleted, but can be marked in-active so there will never be gaps in the Juror ID number.
When Jurors are needed to be added to a Pool, the PRNG is initialed with the seed values.

Jurors that are postponed, deferred, or re-summoned are chosen first and added to the Pool before the randomly selected Jurors.

1. A random number is chosen from 1-N, N=count of Jurors in the Juror table. The random number obtained corresponds directly to the Juror ID - the primary key to the Juror Table.

   The random number chosen is checked against the already chosen Jurors for the Pool and if a duplicate is found, then another number is chosen and compared until a non - duplicate has been found.

2. The Juror record is read from the database
   A. If the Juror is qualified for service they are added to the Pool. To be qualified the Juror record must be marked active and the current Juror Status must be flagged available for service based on the Date Range of the Juror Status and the Date Range of the Pool service dates.
   B. If the Juror is not qualified then another Random number is chosen - Go to Step 1

3. This process repeats until all needed Jurors are added to the Pool.

When Jurors need to be added to a Panel, the PRNG is initialed with the seed values.

1. A random number is chosen from 1-N, N=count of Jurors on the Pool the Panel is pulling from. The random number obtained corresponds directly to the Juror Number on the Pool.

   The random number chosen is checked against the already chosen Jurors for the Pool and if a duplicate is found, then another number is chosen and compared until a non - duplicate has been found.

2. The Juror record is read from the Pool
   A. If the Juror is qualified for service they are added to the Panel. To be qualified the Juror record must be marked active in the Pool and not currently excused in the Pool.
   B. If the Juror is not qualified then another Random number is chosen - Go to Step 1

3. This process repeats until all needed Jurors are added to the Panel.

c# Source Code

Pseudo-Random Number Generator (PRNG) using the Mersenne Twister algorithm by Makoto Matsumoto and Takuji Nishimura. This implementation is rewritten from their C source-code originally dated 2002/1/26. This PRNG has a very long period of 2^19937-1 (approximately 4.3 x 10^6001), and is hence known as MT19937.

```
public class MersenneTwister : RanUInt32
    {
        #region Constructors.
        /// <summary>
        /// Constructs the PRNG-object and seeds the PRNG with the current time of day.
        /// This is what you will mostly want to use.
        /// </summary>
        public MersenneTwister()
            : base()
        {
            Seed();
        }

        /// <summary>
        /// Constructs the PRNG-object using the designated seed.
        /// This is useful if you want to repeat experiments with the
        /// same sequence of pseudo-random numbers.
        /// </summary>
        public MersenneTwister(UInt32 seed)
            : base()
        {
            Seed(seed);
        }

        /// <summary>
        /// Constructs the PRNG-object using the designated array
        /// of seeds. Use this if you need to seed with more than 32 bits.
        /// </summary>
        public MersenneTwister(UInt32[] seeds)
            : base()
        {
            Seed(seeds);
        }
        #endregion

        #region Internal definitions and variables
        static readonly UInt32 N = 624;                     // Array-length.
        static readonly UInt32 M = 397;
        static readonly UInt32 MATRIX_A = 0x9908b0df;       // Constant vector a.
        static readonly UInt32 UPPER_MASK = 0x80000000;     // Most significant w-r bits.
        static readonly UInt32 LOWER_MASK = 0x7fffffff;     // Least significant r bits.
        static readonly UInt32[] mag01 = { 0x0, MATRIX_A };

        UInt32[] mt = new UInt32[N];                        // The array for the state
vector.
        UInt32 mti;                                         // Index into mt-array.

        /// <summary>
        /// Is PRNG ready for use?
        /// </summary>
        bool IsReady = false;
```

```
#endregion

#region PRNG Implementation.
/// <summary>
/// Draw a random number in inclusive range {0, .., RandMax}
/// </summary>
public sealed override UInt32 Rand()
{
    Debug.Assert(IsReady);

    UInt32 y;

    if (mti >= N)
    {
        // Generate N words.

        int kk;

        for (kk = 0; kk < N - M; kk++)
        {
            y = (mt[kk] & UPPER_MASK) | (mt[kk + 1] & LOWER_MASK);
            mt[kk] = mt[kk + M] ^ (y >> 1) ^ mag01[y & 0x1];
        }

        for (; kk < N - 1; kk++)
        {
            y = (mt[kk] & UPPER_MASK) | (mt[kk + 1] & LOWER_MASK);
            mt[kk] = mt[kk + M - N] ^ (y >> 1) ^ mag01[y & 0x1];
        }

        y = (mt[N - 1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N - 1] = mt[M - 1] ^ (y >> 1) ^ mag01[y & 0x1];

        mti = 0;
    }

    y = mt[mti++];

    /* Tempering */
    y ^= (y >> 11);
    y ^= (y << 7) & 0x9d2c5680;
    y ^= (y << 15) & 0xefc60000;
    y ^= (y >> 18);

    Debug.Assert(y >= 0 && y <= RandMax);

    return y;
}

/// <summary>
/// The maximum possible value returned by Rand().
/// </summary>
public sealed override UInt32 RandMax
{
    get { return UInt32.MaxValue; }
}

/// <summary>
```

```
        /// Seed with an integer.
        /// </summary>
        protected sealed override void Seed(UInt32 seed)
        {
            mt[0] = seed;

            for (mti = 1; mti < N; mti++)
            {
                UInt32 lcg = 1812433253;
                mt[mti] = (lcg * (mt[mti - 1] ^ (mt[mti - 1] >> 30)) + mti);
            }

            IsReady = true;
        }

        /// <summary>
        /// Seed with an array of integers.
        /// </summary>
        protected void Seed(UInt32[] seeds)
        {
            Seed(19650218);

            UInt32 i = 1;
            UInt32 j = 0;
            UInt32 k = (N > seeds.Length) ? (N) : ((UInt32)seeds.Length);

            for (; k > 0; k--)
            {
                // Non-linear.
                mt[i] = (mt[i] ^ ((mt[i - 1] ^ (mt[i - 1] >> 30)) * 1664525)) + seeds[j]
+ j;

                i++;
                j++;

                if (i >= N)
                {
                    mt[0] = mt[N - 1];
                    i = 1;
                }

                if (j >= seeds.Length)
                {
                    j = 0;
                }
            }

            for (k = N - 1; k > 0; k--)
            {
                // Non-linear.
                mt[i] = (mt[i] ^ ((mt[i - 1] ^ (mt[i - 1] >> 30)) * 1566083941)) - i;

                i++;

                if (i >= N)
                {
                    mt[0] = mt[N - 1];
                    i = 1;
```

```
            }
        }

        // MSB is 1; assuring non-zero initial array.
        mt[0] = 0x80000000;
    }
    #endregion

    #region Base-class overrides.
    /// <summary>
    /// Name of the RNG.
    /// </summary>
    public override string Name
    {
        get { return "MersenneTwister19937"; }
    }
    #endregion
}
```