



Ranked Choice Voting Tabulation Software Approval

May 2023

Executive Summary

Pursuant to 1VAC20-100-50, the Virginia Department of Elections (ELECT) approves the use of the Universal RCV Tabulator v1.0.1 (RCTab) in Ranked Choice Voting (RCV) contests in the Commonwealth. ELECT has reviewed this software as part of a continuing effort to improve the administration and security of Virginia elections. This memorandum provides a summary of the review of the ELECT Information Security Team completed to ensure the validity of RCTab for use in the Commonwealth.

Background

In the 2020 Virginia General Assembly Session, RCV was established as an optional method for the election of county board of supervisors and city council members.¹ The legislation provided authority to the State Board of Elections (SBE) to promulgate regulations for the proper administration of elections determined by RCV.² The SBE passed RCV regulations at its September 14, 2021 SBE meeting.³ The Arlington County Board of Supervisors adopted an ordinance in December of 2022 approving RCV for use in its primary election in June 2023, the first use of RCV in a local contest.

Voting Systems

RCV allows voter to rank candidates in order of preference. To accommodate this, localities must have the necessary technological capabilities related to their voting systems, which consists of two functions. First, voting systems must have the capability to read ballots with RCV races. After scanning the ballots, voting systems in RCV races must then produce a cast vote record (CVR) of the rankings assigned to candidates on each ballot. Currently, ELECT has approved voting systems from four vendors with these functionalities: Unisyn Voting, Hart Intercivic, ES&S, and Dominion.

Tabulation Software

After a CVR is created by the voting system, additional software is required to tabulate the votes displayed on the CVR based upon the rules prescribed in the RCV regulations. Such software exists both from some voting system vendors and third-party organizations. Currently, voting equipment vendors in Virginia range in their abilities to tabulate results for RCV contests in this second phase. As ELECT was in the process of building policies and procedures to conduct RCV contests throughout the state and evaluating available options for this software, it became imperative to procure the use of a standardized third-party tabulator.

After extensive research into the software best able to provide a uniform method of tabulating RCV contests, ELECT decided to procure the use of RCTab created by the Ranked Choice Resource Center (RCVRC), a division of the Election Administration Resource Center, a nonpartisan 501 (c)(3) nonprofit organization.⁴ ELECT purchased the use of RCTab in January of 2023.

Overview of ELECT's Approval Process

RCTab is a federally tested open-source software that meets the Voluntary Voting System Guidelines (VVSG) established by the U.S. Elections Administration Commission and is the first open-source

¹ [Acts of Assembly, Chapter 1054, 2020 Session](#)

² [§ 24.2-673.1 C](#)

³ [1VAC20-100-10 through 1VAC20-100-80](#)

⁴ RCV Resource Center, <https://www.rcvresources.org/organization>

software to meet VVSG standards.⁵ While the software is open-source, ELECT purchased USB drives containing RCTab directly from the RCVRC. This process aligned with ELECT's security requirements.

In determining whether RCTab was suitable for use in RCV contests, ELECT evaluated the software in three phases:

1. Pro V&V Test Report of Universal RCV Tabulator (RCTab)
2. Review of RCVRC RCTab Security Documentation Package
3. ELECT Test Elections

Pro V&V Test Report of Universal RCV Tabulator (RCTab)

Pro V&V is a voting systems test laboratory accredited by the U.S. Election Assistance Commission for the testing of voting systems to the Voluntary Voting System Guidelines (VVSG 1.0, 1.1 & 2.0).⁶ Pro V&V conducted a test of the Universal RCV Tabulator with voting system vendors to evaluate the certification requirements found in the Voluntary Voting System Guidelines Version 1.0 (VVSG 1.0).⁷

Various operating systems were used in Pro V&V's test. Hash values were recorded to ensure the software version being used matched the original. RCVRC's technical data package (TDP) contained the information needed to conduct the tests. Of note, "[t]he security testing evaluated the effectiveness of the Universal RCV Tabulator in detecting, preventing, recording, reporting, and recovering from security threats. To evaluate the integrity of the Universal RCV Tabulator attempts were made to defeat the access controls and security measures documented in the system TDP."⁸ All tests were passed.

Review of RCVRC RCTab Security Documentation Package

ELECT's next reviewed the RCVRC security documentation package to verify that it aligned with ELECT's pre-established security procedures. ELECT's Security Division reviewed a collection of documents provided by the RCVRC in preparation for ELECT's RCTab approval. The documents reviewed are summarized as follows and included in the Appendix:

- **Secure USB Processes.**⁹ The Secure USB Process explains RCVRC's flash drive security features and attributes such as AES 256 bit encryption, FIPS 140-2, level 3 standard and PIN authenticated hardware encryption. These meet federal government standards.¹⁰ Thorough instructions are included to set and update the security PIN preventing tampering or alteration of the program.
- **System Hardening Procedures.**¹¹ System hardening procedures are focused on hardening a machine to run the software. Like most security procedures, removing unnecessary programs is

⁵ Ranked Choice Voting Resource Center, RCTab, <https://www.rcvresources.org/rctab>

⁶ See Appendix 1

⁷ See Appendix 2

⁸ *Id.* at 6

⁹ See Appendix 3

¹⁰ NIST Digital Identity Guidelines – Authentication and Lifecycle Management, <https://pages.nist.gov/800-63-3/sp800-63b.html>

¹¹ See Appendix 4

important to reduce the attack surface. Software that is loaded includes: Windows 10 with latest service pack, antivirus, Windows updates, Windows Defender, Excel and limited drivers. Since the machine is offline, the software is installed and updated via a separate USB. Step by step instructions are provided to complete all the updates. All physical ports are sealed except for power and one USB. Additional hardening settings are given which further block potential hardware and software gaps. The result is a system secured far beyond standards that never goes online. An uninterruptible power supply (UPS) is connected to ensure no power loss occurs during processing. Lastly, the system is only accessible to authorized users and stored in secured area.

- **Software Design and Specifications.**¹² The Software Design and Specifications explains the code used to run the application. RCVRC software design and coding standards are based on the VVSG 1.0 VOL 1: 5.2. Additionally, they use Google Checkstyle to check code quality. The tabulator runs on Java. The code is open source for transparency. Java classes run the needed functions for the various election system vendors. There are also 3rd party modules that meet VVSG requirements. They are original, unmodified and again open source. Several scenarios are coded to ensure accurate outcomes. Logging captures exceptions. All of this is done offline on the tabulation system.
- **System Test and Verification Specification.**¹³ The System Test and Verification Specification outlines the battery of tests RCTab goes through. Testing must meet V.2:2.7.2 National Certification Test Specifications and benchmarks for Acceptance criteria, Processing accuracy, Data quality assessment and maintenance, Ballot interpretation logic, Exception handling, Audit and Security.
- **RCTab Logic & Accuracy (L&A) Test Procedures.**¹⁴ The RCTab Logic & Accuracy (L&A) Test Procedures verify the application is correctly configured and operating. L&A is always performed conducted prior to use in an election. A secure USB and laptop frame this testing. From there it's a step by step. User are instructed to create hash files for the results. This is best practice to ensure integrity (no alterations were made to the files).
- **Quality Assurance Plan.**¹⁵ The Quality Assurance Plan outlines the processes and procedures of the RCTab software. The system must meet the prerequisites (including no wireless). The components were chosen in accordance with VVSG Volume 2:2:12.1 as it refers to Volume 1:8.5 and Volume 1:8.6. Test data storage process has been updated to meet the specifications outlined in the VVSG Volume 1:8.5c. Quality testing includes: defects, regression, various voting system vendors, large contests and various "Real life" scenarios. Defects are mitigated according to criticality and can be tracked at the Brightspots/RCV Github site. This tracking keeps transparency and helps the community stay informed.

¹² See Appendix 5

¹³ See Appendix 6

¹⁴ See Appendix 7

¹⁵ See Appendix 8

ELECT Test Elections

Testing was conducted by ELECT using various sample scenarios in mock elections to confirm the compatibility of RCTab with voting systems currently approved in Virginia. Further, ELECT will require validation testing and confirmation prior to each RCV contest. The mock elections were conducted on the following dates:

Dominion	1/23/2023
ESS	1/13/2023
Hart	1/23/2023
Unisyn	2/22/2023

ELECT staff tested for instant runoff (single winner) and single transferable vote (multi-winner) races for each vendor.

ELECT Approval of Software

In summary, based upon the information reviewed and the processes and procedures outlined in the attached materials, ELECT's security department concludes that the RCTab software provides a secure solution to tabulate RCV elections.

Appendix

1. Pro V&V Certificate of Accreditation from United States Election Assistance Commission
2. Pro V&V Universal RCV Tabulator Test Report
3. Secure USB Processes
4. System Hardening Procedures
5. Software Design and Specifications
6. System Test and Verification Specification
7. RCTab Logic & Accuracy (L&A) Test Procedures
8. Quality Assurance Plan

Appendix 1



United States Election Assistance Commission

Certificate of Accreditation

Pro V&V, Inc.
Huntsville, Alabama

is recognized by the U.S. Election Assistance Commission for the testing of voting systems to the Voluntary Voting Systems Guidelines VVSG 1.0, 1.1 & 2.0 under the criteria set forth in the EAC Voting System Testing and Certification Program and Laboratory Accreditation Program. Pro V&V is also recognized as having successfully completed assessments by the National Voluntary Laboratory Accreditation Program for conformance to the requirements of ISO/IEC 17025 and the criteria set forth in NIST Handbooks 150 and 150-22.

Date: 12/21/22

Original Accreditation Issued on: 2/24/2015

*Accreditation remains effective until revoked
by a vote of the EAC pursuant to 52 U.S.C. §
20971(c)(2).*

Mark A. Robbins
Interim Executive Director, U.S. Election Assistance Commission
EAC Lab Code: **1501**

Appendix 2

Letter Report



To: The Election Administration Resource Center dba Ranked Choice Voting Resource Center

From: Wendy Owens - Pro V&V, Inc.

CC: Keith Long – RCVRC; Alan Simmons, Michael Walker - Pro V&V, Inc.

Date: August 30, 2019

Subject: Universal RCV Tabulator v1.0.1

Dear RCVRC:

Pro V&V, a VSTL, is providing this letter to report the results of the modification testing performed on the RCVRC's Ranked Choice Voting Tabulator (Universal RCV Tabulator 1.0.1), a post-EMS secondary tabulator, which in this instance is an external software modification-addition to the ES&S's EVS 6.0.4.0 EAC Certified Voting System (ESSEVS6040 – May 03, 2019).

This testing campaign was performed, with the support of ES&S, to verify that the submitted modification-addition meets the certification requirements found in the Voluntary Voting System Guidelines Version 1.0 (VVSG 1.0).

To start, Pro V&V determined that this modification was subject only to limited certification testing as RCVRS was able to establish that this detached modification-addition did not affect the previously demonstrated compliance of the baseline system to the VVSG 1.0. Limited testing, in addition to other stated uses, is intended to facilitate the integration of vote counting software with other systems and election software. The following modifications to the previously certified system were evaluated:

- Addition of the Universal RCV Tabulator

Background

The ES&S EVS 6.0.4.0 voting system is a complete EAC certified voting system composed of EMS software applications, central count devices, and polling location devices.

This Universal RCV Tabulator can process data from voting machines that are capable of exporting cast vote records (CVR's) and tabulate a single-winner or multi-winner ranked choice voting election according to the rules used in current state, county or city election jurisdiction in the United States. The Universal RCV Tabulator outputs the tabulation results in comma separated values (.csv) tabular data format and creates an audit file for the RCV election.

Although the Tabulator was tested using files produced by the EVS 6.0.4.0, ES&S confirmed that the exported files created from all EVS certified systems among EVS 5.0.0.0 thru 6.0.4.0 are identical and would produce the same testing results with the Universal RCV Tabulator as noted in this report.

Application Summary

The Universal RCV Tabulator was configured and setup as it would be for normal operations as per the TDP. Both RCVRC and ES&S provided materials and phone support as needed to aid in the testing process.

For testing, the Universal RCV Tabulator resided on three independent test support platforms with the Cast Vote Records (CVR) originating from the ES&S EVS 6.0.4.0 baseline system. The modified system supported voting variations are:

- Closed primaries
- Open primaries
- Partisan offices
- Non-partisan offices
- Write-in voting
- Ballot rotation
- Straight party voting
- Cross-party endorsement
- Split precincts
- Vote for N of M
- Ranked order voting (w/Universal RCV Tabulator)
- Provisional or challenged ballots

Software

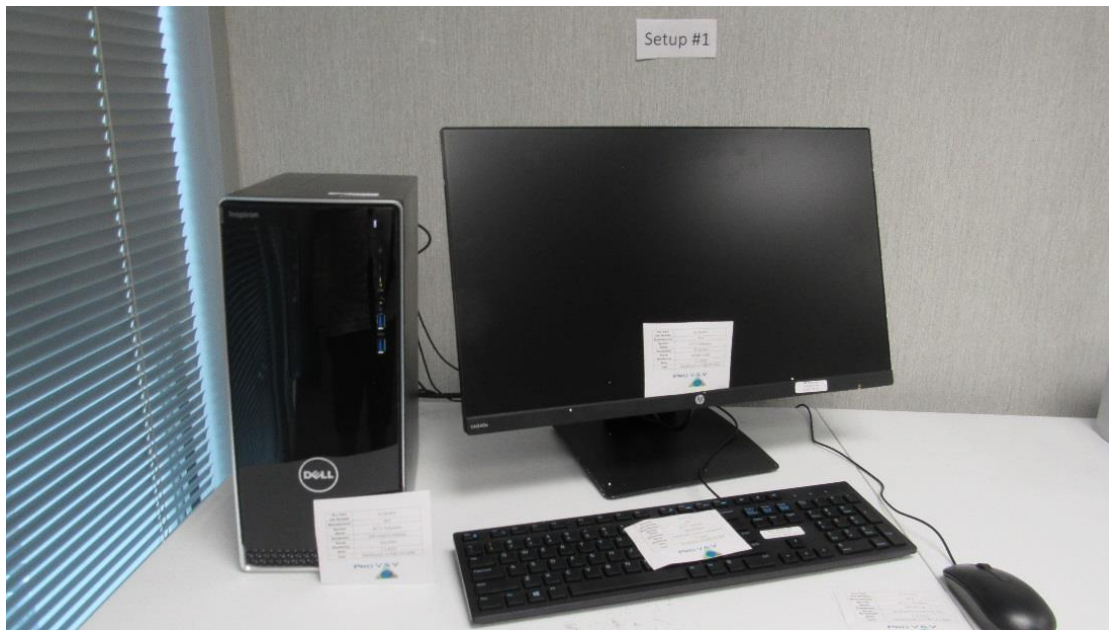
Component Name	Version	Unique Identifier (SHA 512 hash value)
Universal RCV Tabulator	Windows 1.0.1	a83d84e22453ac656d9fd1b78e326e96a034801174b043a4ba1afa5167442050059d74962e6e0ee278063797527aa009d33ea70951e37fc30cc3cc77163800da
Universal RCV Tabulator	OSX 1.0.1	a0ea658cd0d8b685436aec7a240de3f9b0ed84980823a489586c249a79c1da8a4f5bfc99f94c10969ebb821f53dbb0d9c6da606b28adee8cfe4fa7e3beab4f7b
Universal RCV Tabulator	Linux 1.0.1	92c7af9b074929206e196e711c7399c51f8386712b6f1b7b5c31e51be18511c8af7cfca36c77427ed832f89b8b186d59f97f2cdccbc78e990a2dab7da1f2c5b4

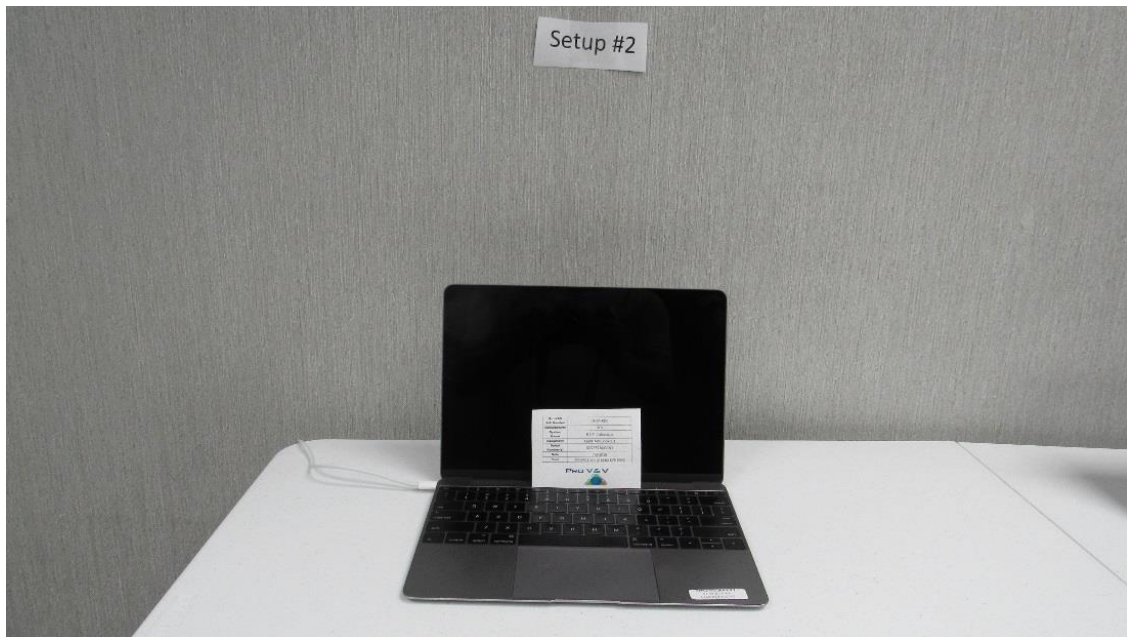
Testing Hardware

Component Name	Model/Version Number	Serial Number(s)	Description
Universal RCV Tabulator	Dell Inspiron Desktop Windows 10 Pro	1QLJDW2	Universal RCV Tabulator Platform
Universal RCV Tabulator	Apple Macbook 8.1 Mac OS 10.13	CO2PP2BAGCN3	Universal RCV Tabulator Platform
Universal RCV Tabulator	HP Elite Book Laptop 840 Mint 19.1	SCG6284VLW	Universal RCV Tabulator Platform
Printer	Samsung Xpress M2020W Printer	CNB3M47DJ8	Universal RCV Tabulator Printer
Uninterruptable Power Supply	APC Backup UPS 600	4B191P18483	Universal RCV Tabulator External Power Supply

Testing Support Materials

Material	Quantity	Product Number(s)	Description
8.5 x 11 Printer Paper	100 Sheets	HP 200350	Report Paper
Thumb Drive	4	N/A	Transport Media

**Photograph 1: Universal RCV Tabulator Setup 1 (Windows)**



Photograph 2: Universal RCV Tabulator Setup 2 (OSX)



Photograph 3: Universal RCV Tabulator Setup 3 (Linux)

Testing Summary

This testing campaign concentrated on the addition of a Universal RCV Tabulator to the ES&S EVS 6.0.4.0 EAC certified voting system and the associated characteristics that necessitate further review.

Technical Documentation Package (TDP) Review

The TDP review was done to verify that the submitted Universal RCV Tabulator documentation contained sufficient technical and administrative information to allow successful operation of the Universal RCV Tabulator with the baseline system. The TDP review also assessed the extent to which the modified TDP adhered to the VVSG 1.0. This evaluation was an iterative process with concerns communicated to the RCVRC as early in testing as possible for resolution. All changes made to the TDP during testing were subjected to re-examination.

RCV TDP

Document #	Document	Version #
000	Technical Data Package	0.3
100	Tabulation Options for RCV	0.1
105	Overview	0.1
110	Expected Outcome RCV Test Sets Single-Winner	0.1
115	Expected Outcome RCV Test Sets Multi-Winner	0.1
125	ES&S Ballot Layout Limits	0.2
130	RCV Tabulator User Guide Overview	0.3
150	Universal RCV Tabulator System Security Specifications	0.3
120	Installation Instructions for RCV Tabulator	0.7
121	Installation Instructions for Universal RCV Tabulator – MacOS	0.4
122	Installation Instructions for Universal RCV Tabulator – Linux	0.4
140	RCV Tabulator Operation Instructions	0.4
141	Universal RCV Tabulator Operation Instructions	0.4
142	Universal RCV Tabulator Operation Instructions	0.4
135	Configuration File Parameters	0.4
200	Universal RCV Tabulator Election Setup Checklist - MANUAL	0.3
201	Universal RCV Tabulator Election Setup Checklist - INTERACTIVE	0.3
205	Universal RCV Tabulator cvr Files	0.1
210	Universal RCV Tabulator Election Data Form	0.3
215	Universal RCV Tabulator cvr Files	0.1
220	Universal RCV Tabulator Election Rules	0.3
230	Universal RCV Tabulator Logic & Accuracy Test	0.1
250	Universal RCV Tabulator Election Processing	0.3
251	Universal RCV Tabulator Election Processing Linux OS	0.1
252	Universal RCV Tabulator Election Processing MacOS	0.1
225	Universal RCV Tabulator System Operator Log Errors	0.2

Summary Findings: *The addition of the Universal RCV Tabulator TDP did not affect the ability of the baseline TDP to satisfy all VVSG 1.0 requirements. Additionally, the final Universal RCV Tabulator documentation editions provided sufficient detail to adequately attend to the added functionality.*

Physical Configuration Audit (PCA)

The PCA was conducted to ensure that the configurations and setups of the Universal RCV Tabulator submitted for testing matched the vendor's technical documentation, and confirmed that the documentation was sufficient for the user to install, validate, operate, and maintain the Universal RCV Tabulator. All changes made to the Universal RCV Tabulator during testing were subjected to re-examination.

The PCA further established a configuration baseline of the software and hardware to be tested.

Summary Findings: *The addition of the three employed stand-alone and isolated Universal RCV Tabulator platforms matched the documentation and did not impact the ability of the baseline system as modified to satisfy the VVSG 1.0 requirements.*

Functional Configuration Audit (FCA)

The FCA testing utilized standardized test cases to verify that the Universal RCV Tabulator satisfied prescribed requirements and performed the functions claimed in the system documentation.

Summary Findings: *The Universal RCV Tabulator passed all of the applicable pre-vote, vote, and post-vote test case requirements. The Universal RCV Tabulator demonstrated the ability to function as required and intended, and does not impact the ability of the baseline system as modified to satisfy the VVSG 1.0 requirements.*

Source Code Review

The Universal RCV Tabulator source code review was to ascertain how completely the software conforms to the vendor's specifications. The source code inspection also assessed the extent to which the code adhered to the VVSG 1.1.

Summary Findings: *The Universal RCV Tabulator stands as a VVSG 1.1 compliant source code modification to a VVSG 1.0 system, and meets all of the VVSG 1.1 source code requirements.*

Security Testing

The security testing evaluated the effectiveness of the Universal RCV Tabulator in detecting, preventing, recording, reporting, and recovering from security threats. To evaluate the integrity of the Universal RCV Tabulator attempts were made to defeat the access controls and security measures documented in the system TDP.

Summary Findings: *A fundamental security review was performed on the Universal RCV Tabulator, which incorporates the security policies of the baseline system.*

The Universal RCV Tabulator was found to have an applied level of security that complies with the verified VVSG 1.0 security provisions. Additionally the presence of the Universal RCV Tabulator does not impact the ability of the baseline system as modified to satisfy the VVSG 1.0 security requirements.

Usability Testing

The usability testing encompassed a range of methods that examined how users in the target audience actually interact with the Universal RCV Tabulator.

Summary Findings: *The Universal RCV Tabulator has operations and controls that can easily be employed by users in the target audience making use of the system TDP.*

Accessibility Testing

The accessibility testing measured characteristics that indicated the degree to which the Universal RCV Tabulator is available to, and usable by, users in the target audience.

Summary Findings: *The Universal RCV Tabulator is accessible to users with a basic understanding of computers and election management.*

System Integration

The system level testing addressed the integrated operations of the Universal RCV Tabulator with the baseline system, and tested the Universal RCV Tabulator capabilities as part of the baseline system as a whole. This testing validated the seamless functional integration and data inputs/outputs of the baseline system with the Universal RCV Tabulator. For this testing the Universal RCV Tabulator were configured and operated as for normal operations.

Summary Findings: *Cast Vote Records (CVR) that were generated by the baseline system were imported into the Universal RCV Tabulator and processed through tabulation to report generation. No system issues were noted during testing.*

Accuracy Testing

Universal RCV Tabulator accuracy was based upon the totality of all testing.

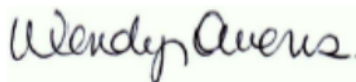
Summary Findings: *During testing, actual results were validated against expected results. No accuracy issues were noted throughout the test campaign.*

Conclusion

Based on the testing performed and the results obtained, the Universal RCV Tabulator solution identified in this letter-report is believed to meet the requirements set forth by the VVSG 1.0.

Should you require additional information or would like to discuss this matter further, please contact me at 256-713-1111.

Sincerely,

A handwritten signature in dark ink that reads "Wendy Owens". The signature is written in a cursive, flowing style. The name "Wendy" is written in a larger, more prominent script than "Owens".

Wendy Owens
VSTL Program Manager
wendy.owens@provandv.com

Appendix 3

RCTab v.1.2.0 - Secure USB Process v.1.0.0

RCTab is available as a download from the RCVRC website or directly from Github, where each build of the software is hosted. However, jurisdictions are free to request a copy of the software on a secure USB.

The steps below outline the details of how the RCVRC selects and secures a USB for use and transport to the requestee.

USB Details

Security begins with the USB storage device selected. Currently, the RCVRC uses iStorage datAshur PRO encrypted USB Memory Sticks. iStorage's datAshur PROs were selected because they include the following features and attributes:

- PIN authenticated hardware encryption
- AES-XTS 256-bit hardware encryption
- Certified to meet NIST's FIPS 140-2 Level 3 standard
- Compatible with MS Windows, macOS, and Linux

For more information on iStorage's datAshur PROs functionality please see the appendix at the end of this guide.

USB Formatting

Each datAshur PRO memory stick arrives formatted by the factory. However, the RCVRC formats each USB memory stick in-house to ensure security. Formatting is done using Windows 11's disk formatting tool.

To format a datAshur PRO memory stick, the RCVRC follows the following steps:

1. Unlock the datAshur PRO memory stick by pressing the key icon button on the memory stick and entering the default admin password, 1-1-2-2-3-3-4-4. Press the key icon button again, and the LED above the keypad will turn green.
2. Plug the datAshur PRO memory stick into a USB slot on your PC.
3. Select the search icon on your window's toolbar, which is typically located in the bottom left-hand corner of your screen, and type "File Explorer".
4. Once File Explorer is open, select "This PC" on the left-hand side of the screen.
5. Under Devices and drives, right-click on the USB memory stick and then select "Format".
6. In the Format USB Drive window, leave each option alone except for "Format Options." Unselect "Quick Format."
7. A warning prompt will pop up. Click "Ok"
8. Formatting complete.

Setting a PIN on datAshur PRO Memory Stick

Once the datAshur PRO Memory Stick has been formatted, the manufacturer will set a new user PIN is programmed into the device. This PIN will be shared with the organization or jurisdiction requesting the USB version of RCTab. User PIN requirements are as follows:

- Must be between 7-15 digits in length
- Must not contain only repetitive numbers, e.g. (3-3-3-3-3-3)
- Must not contain only consecutive numbers, e.g. (1-2-3-4-5-6-7)
- Must not be plugged in a USB port at the time of updating

To update a pin on the datAshur PRO Memory Stick follow these steps:

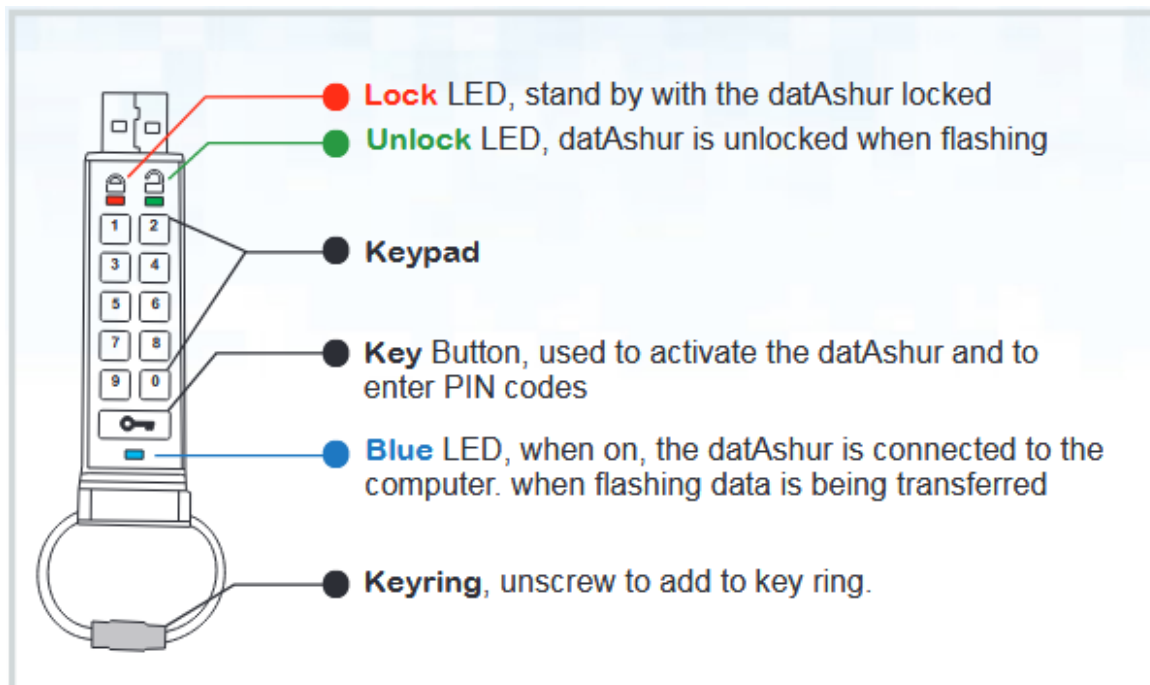
1. Press the key button and wait for the red LED to blink.
2. Enter the user PIN and press the key button. The red LED will be solid for two seconds and then change to a solid green LED light.
 - a. The factory default PIN is 1 1 2 2 3 3 4 4
3. Press the key button twice and the blue LED will blink.
4. Within 10 seconds, enter a new user pin and press the key button twice. The green LED will blink after entry.
 - a. Decide on a PIN first and write it down before trying this step.
 - b. The new PIN can be a word and typed using the alpha-numeric keypad on the datAshur PRO Memory Stick.
5. Re-enter the new user PIN and press the key button twice. The red LED will illuminate and then change to a solid green LED if PIN entries match.
6. Once the datAshur PRO Memory Stick locks, enter the new PIN and verify it unlocks the memory stick.
 - a. After the PIN has been changed, the default PIN 1 1 2 2 3 3 4 4 will no longer work.

Preparing datAshur PRO Memory Stick for a jurisdiction

After a datAshur PRO Memory Stick has been formatted and the PIN has been reset, the requested version of RCTab and any relevant documentation will be loaded onto the memory stick. The details of what is stored on the memory stick, as well as the PIN, will be shared with the organization or jurisdiction.

Appendix

1. datAshure PRO Layout



2. LED Indicators and their actions

LED	LED State	Description	LED	LED State	Description
	Red - Fade Out 	Locking down/incorrect PIN entry		Red and Green blinking alternately 	Factory reset/deleting files in Admin mode
	Red blinking 	Locked and awaiting factory default PIN or User defined PIN entry		Red and Green flickering together 	Awaiting Admin PIN entry
	Green Solid 	datAshur PRO is unlocked in User mode		Green and Blue blinking together 	User Options mode
	Green blinking 	When connected to a USB port if Green Led blinks every 2 seconds this indicates the datAshur PRO has been set as 'Read-Only'		Green and Blue flickering together 	Admin Options mode
	Green flickering 	datAshur PRO is unlocked in Admin mode		Red and Blue blinking together 	When not connected to a USB port indicates that both User and Admin PINs have been set on the datAshur PRO
	Blue Solid 	Connected to a USB port		Red and Blue flickering together 	Awaiting Admin PIN change
	Blue Blinking 	Data exchange with host/changing User PIN/when not connected to a USB port indicates an Admin PIN exists		Red, Green & Blue blinking in sequence to Red and fade out 	Drive has developed a fault. Please retry the command and if the problem persists contact technical support

Document Revision History

Date	Version	Description	Author
01/06/2023	1.0.0	Document Created	Rene Rojas

Appendix 4

System Hardening Procedures

URCVT v. 1.2.0 16 System Hardening Procedures v. 1.1.1

Any recommendations listed in this document should not supersede user jurisdiction procedures or other controlling governance entities.

The hardware used to house the Universal Ranked Choice Voting Tabulator, URCVT, software should, at minimum, follow the steps below to ensure the hardware is adequately protected against unauthorized access, theft of data, and/or malicious attacks.

Hardening of the operating system (Windows 10) is a way to make the computer and data more secure. It may include removing unused applications and files, establishing password login protection on Windows 10, disabling automatic windows login, keeping the system updated appropriately, proper configuration of the system, applying patches, and security updates, among other things. All URCVT systems must include only the following software:

- Windows 10 Pro with the latest service pack installed
- URCVT v. 1.2.0
- Microsoft Excel 2007 or above
- Users must also retain access to:
 - Command Prompt
 - Digital Signature tools
 - Decryption tools, such as BitLocker
 - Notepad
- Antivirus software
- UPS and printer drivers

See pages 8-9 below for procedures for securing verified versions of COTS software listed above.

Note that none of the steps below requires the presence of any special environment variables.

In addition to the system hardening protocols, it is recommended that only authorized users (no less than two bipartisan employees) should have physical access to the standalone computer. The computer should be in a secure location.

Windows Update Procedures

Preparing Offline Updates

URCVT should run on a workstation that is in a closed environment (not on a network) without access to the internet. Hardware drivers, updates, and virus protection should be downloaded on another computer and transferred to the workstation by a removable device such as a USB

flash drive. Prepare for offline updates by downloading the following to the USB drive designated for use for this purpose: Hardware Drivers, Windows Updates, Windows Defender Offline Updates.

Hardware Drivers

- Locate the computer identification information provided by the manufacturer.
- On a second internet-connected computer, locate the appropriate hardware drivers necessary to complete the computer setup.
- Download the .cab driver package and extract it.
- Copy the extracted files to a USB drive. Keep the USB drive for the next procedure.

Windows Updates

- On a second machine connected to the Internet, go to <http://download.wsusoffline.net/> and download the latest WSUS Offline Update version.
- Extract the downloaded archive.
- Navigate to the extracted archive and double-click on UpdateGenerator.exe.
- Verify the following settings are checked:
 - Under Windows 10x64 versions, select:
 - Your current version of Windows 10
 - Under Options, select:
 - Verify downloaded updates
 - Use 'security only updates' instead of 'quality rollups.'
 - Include C++ Runtime Libraries
 - Under USB medium, select:
 - Copy **updates for selected products into the directory**, and next to it, set the directory where the program will download the updates.

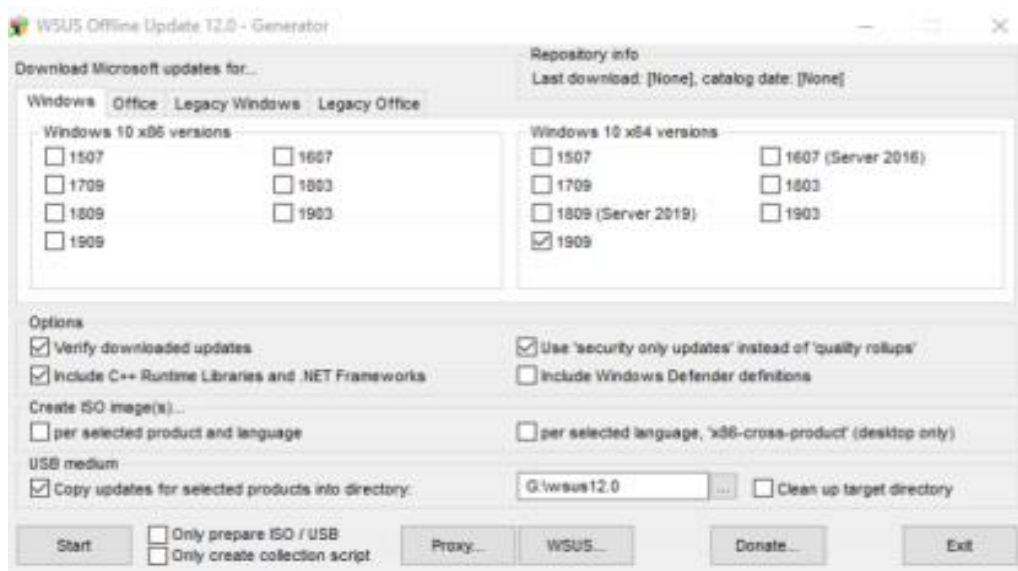


Figure 2-1: WSUS Offline Updater Settings

- Click **Start**.
- If the version check window appears asking to update WSUS, click **NO**. WSUS

- opens several command-line windows to perform operations.
- When the complete dialog window appears asking if you would like to review the log file, click **NO**.
- Copy extracted files to a USB drive. Keep the USB drive for the next procedure.

Windows Defender Offline Updates

- On a second computer connected to the Internet, go to the following website:
<https://www.microsoft.com/en-us/wdsi/definitions>
- Find the link for Windows Defender in Windows 10 (64 bit). Click the link to download the update.
- Transfer the downloaded file to a USB drive and keep it for later installation.
As a note, all USB drives used for these purposes should be kept in a safe place in manufacturer recommended conditions.

Configuration of Windows Updates

This section covers the configuration of the operating system and hardware drivers. Since the workstation is not connected to the internet, updates and security patches must be installed manually.

Note: Depending on the build of your Windows 10, the UI might be slightly different than described in the section below. This document follows Windows Pro 10, build 1909. You should use your current version of Windows 10.

Drivers and Tools

Proper drivers may be required for each device needing to be installed. This depends on the hardware that you are using.

- At the non-internet connected workstation, connect the USB drive prepared in the section above, titled *Preparing for Updates*.
- Click **Start**, search **Device Manager**, and open it.
- In **Device Manager**, expand the **Other Devices** section.
- Right-click the first driver with a yellow icon and select **Update Driver Software**.

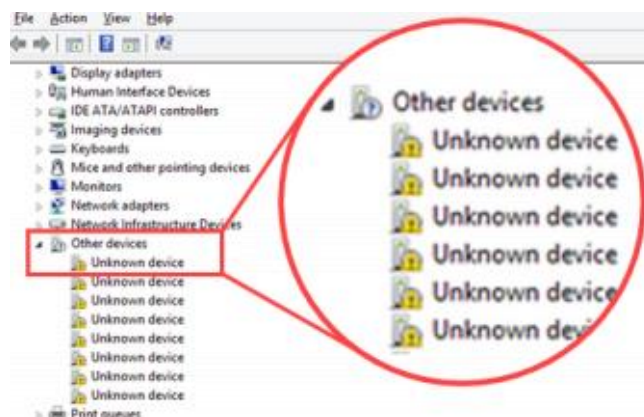


Figure above: Unknown devices in Device Manager

- Follow the prompts.
- When asked, “How do you want to search for driver software,” select **Browse my computer for driver software**.
- Click **Browse**.
- On the Browse for Folder dialog, navigate to the extracted driver files on the USB drive.
- Select the folder containing the drivers and click **OK**.
- On the Update Drivers dialog, click **Next** to run the update.
- When you see a confirmation message, click **Close** and repeat the steps for every driver with a yellow icon. Not necessary for USB Device.

Updating Windows

Updates to Windows 10 are vital to maintaining a system that is secure and optimized. These updates include security patches and updates. Since the URCVT workstation operates in a closed environment that is not connected to the internet, WSUS Offline Update is an alternate tool that must be used to install the updates on the standalone workstation. The tool was prepared in the section above, *Preparing Offline Updates*.

- At the URCVT workstation, connect the USB drive prepared in the above section.
- Navigate to the USB drive and run **UpdateInstaller.exe**.
- Verify the following boxes are selected:
 - Update C++ Runtime Libraries
 - Update Root Certificates
 - Verify installation packages
 - Show log file

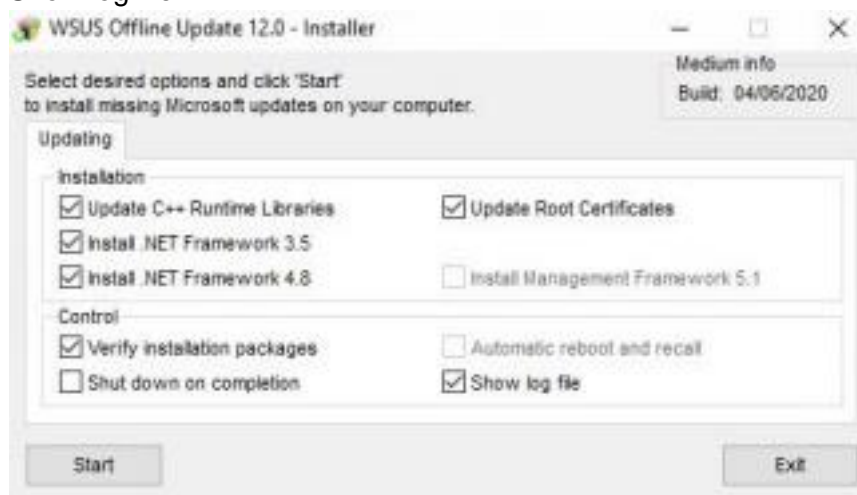


Figure above: WSUS Offline Update Installer Settings

NOTE: A warning window appears when Automatic reboot and recall is checked. Click **Yes** to disable User Account Control (UAC) temporarily. WSUS cannot operate unless UAC is disabled.

- Click **Start** to begin installing updates.
- When the cmd window prompts you to reboot & recall the system, restart the machine and log back in.

- Navigate to the WSUS installer folder and run: **UpdateInstaller.exe**.
- Click **Start** on the installer to continue offline updates installation.
- Repeat reboot and recall until the cmd window prompts you to only **Reboot the machine**.

Additionally, the following items should be downloaded manually from the second internet-connected computer. Download to the USB drive and install on URCVT computer that is operating in a closed environment and that is not connected to the internet.

Please note that the following Windows Security Updates are meant to be examples ONLY. Jurisdictions should always check for the most up-to-date Windows 10 updates available from Microsoft.

2020-03 Cumulative Update for Windows 10 Version 1909 for x64-based Systems (KB4554364):

- Copy and paste the following web link to your browser:
<https://www.catalog.update.microsoft.com/Search.aspx?q=KB4554364>
- Next to: *2020-03 Cumulative Update for Windows 10 Version 1909 for x64-based Systems (KB4554364)*, press **Download**. A *Download* window displays.
- Click on the *.msu* link. The file starts to download.

2020-01 Servicing Stack Update for Windows 10 Version 1909 for x64-based Systems (KB4541338):

- Copy and paste the following web link to your browser:
<https://www.catalog.update.microsoft.com/Search.aspx?q=KB4541338>
- Next to: *2020-03 Servicing Stack Update for Windows 10 Version 1909 for x64-based Systems (KB4541338)*, press **Download**. A *Download* window displays.
- Click on the *.msu* link. The file starts to download.

Installing Offline Updates for Windows Defender

To update Windows Defender:

- Close the Windows Defender.
- Connect the USB drive prepared in the section above.
- On a non-internet connected workstation, double click the *mpam* file to install it. There will be no prompts or installation process.
- To verify the update was applied, open Windows Defender, locate *Virus and Spyware Definitions*. It would say *Up to date* if the installation was successful.

Disabling Automatic Updates

Windows will use resources attempting to update Windows 10. Windows Update must be disabled to prevent the system from using those resources.

To disable automatic updates:

- From the Start Menu, enter gpedit.msc. gpedit.msc appears in the search results.

- Launch **gpedit.msc**.
- From the Local Group Policy Editor, expand the following nodes: **Computer Configuration > Administrative Templates > Windows Components**.
- Select **Windows Update**.
- From the Details pane, double-click **Configure Automatic Updates**. 6. Select **Disabled**, then click **OK**.
- Close the Local Group Policy Editor.
- From the Start Menu, enter CMD. Command Prompt appears in the search results.
- Right-click **Command Prompt** and then select **Run as Administrator**. If the UAC dialog appears, click **Yes**.
- In the Administrator Command Prompt window, enter the following two commands, each followed by enter.
 - `sc config wuauserv start=disabled`
 - `NET STOP wuauserv /y`
- Close Command Prompt.

Windows Security Procedures

With the non-internet connected computer now fully installed, security measures should be applied. Measures such as device encryption and hardening of the Operating System will help keep the integrity of the system and any data that is processed on the workstation. Step-by-step instructions for hardening the system are provided below.

NOTE: Before applying security settings, make sure all system and third-party components are installed and configured. After security settings are applied, some earlier steps may be impossible to perform due to the hardened state of the Operating System. Once applied, security settings cannot be undone.

Device Encryption

Device encryption protects your data on your device. With device encryption, only authorized people can access the device. Device encryption is available on most devices. If not, standard BitLocker encryption may be allowable instead. For information about device encryption on Windows 10 and step by step processes for:

1. Determining if you can use the device encryption or should use BitLocker standard encryption.
2. How to turn on device encryption or BitLocker standard encryption.
3. How to obtain a recovery key.

Go to the Microsoft Windows 10 Device Encryption page at: <https://support.microsoft.com/en-us/windows/device-encryption-in-windows-10-ad5dcf4b-dbe0-2331-228f-7925c2a3012d>.

Physical Security and Hardening

Users must physically seal all external ports on hardware URCVT is installed on, except ports used for power supply, necessary external displays, and one (1) USB port. The user jurisdiction should employ a policy where the use of tamper-evident and tamper-resistant seals are used to identify ports that should never be accessed, unlikely to be accessed, and can be accessed if necessary.

Operating System Hardening

Operating system hardening can reduce the risk of a security breach by removing or disabling many non-essential software and hardware components that could act as a back door for

attackers. The vendor highly recommends the following hardening procedures to protect your standalone computer.

Disable Network Connection From Network Connection Settings:

- 1) Press Win+R and enter *ncpa.cpl* to open the Network Connection window.
- 2) For each network connection, right-click on it:
- 3) Select disable.

Set ScreenSaver Password:

- 1) Open the Control Panel.
- 2) Click Appearance and Personalization.
- 3) Click Change screen saver.
- 4) In the Screen Saver settings, check the box *On resume, display logon screen*.

Disable Automatic Login:

- 1) Press Win+R, enter *netplwiz* to open the *User Accounts* window.
- 2) Check the option for *Users must enter a username and password to use this computer* and click Apply.

Disable Remote Access:

- 1) Type *remote settings* into the search box.
- 2) Select *Allow remote access to your computer* to open the Control Panel for Remote System Properties.
- 3) Check *Don't Allow Remote Connections* to this computer.

Enable Firewall:

- 1) Open the Control Panel in Windows.
- 2) Click on System and Security.
- 3) Click on Windows Firewall.
- 4) In the left navigation pane, click *Turn Windows Firewall On*.

Disable all network interfaces:

- 1) Enter cmd in the windows search bar.
- 2) *Command Prompt* application shows up in search results.
- 3) Right-click on the *Command Prompt*.
- 4) Select *Run as administrator*.
- 5) Type "C:\Windows\System32\netsh.exe interface show interface" and press Enter.
- 6) For each network device listed, enter the following command, replacing "Interface Name" with each of the names returned from step 5 (include the double-quotes):
C:\Windows\System32\netsh.exe interface set interface "Interface Name" disable

Global Disclaimer for non-URCVT software processes and procedures:

Always refer to the manufacturer's current documentation and recommendations for the latest and most secure manner to download, store, and verify the installation of software

dependencies. If questions or issues arise, consult with an information technology security professional for additional information and assistance.

Verify Microsoft Excel installation:

Microsoft Excel Installation:

- Download Microsoft Office on a different computer at <https://support.microsoft.com/en-us/office/use-the-office-offline-installer-f0a85fe7-118f-41cb-a791-d59cef96ad1c>.

Storing Microsoft Excel Installer:

- Generate an SHA-512 Checksum using the following commands if the download source does not provide one:
 - C:\Windows\System32\certutil.exe -hashfile [filename] SHA512
 - Store this checksum in a separate location, to be used for verification later
- Copy the downloaded .exe file to an external hard drive

Installing Microsoft Excel:

- Connect the external hard drive to the standalone URCVT computer.
- Copy the downloaded .exe file from the external hard drive to the URCVT computer.
- Generate an SHA512 Checksum using C:\Windows\System32\certutil.exe -hashfile [filename] SHA512.
- Compare the SHA512 hash code produced on your system to the SHA512 hash code generated on the downloading computer or with the SHA512 hash from the download source.
- If the hashes match, continue with the installation of the executable downloaded file.

Verification:

- Open Microsoft Excel from the "Start" menu.
- If Excel opens, Microsoft Excel is installed.

Verifying Antivirus is Installed (Optional):

Antivirus software is typically recommended for all Windows systems. Using software such as Webroot can serve this purpose.

Webroot Installation (or alternate antivirus software)

- Purchase Webroot by going to <https://www.webroot.com/us/en/business/smb/endpoint-protection> on a separate computer. To purchase alternative antivirus software, use trusted sources for purchase and download.
- Save antivirus software to an external device.
- Connect the external device to the standalone URCVT computer.
- Follow all manufacturer instructions for installation and verification.

Verifying Printer Drivers:

- Download any printer drivers to a different computer and save them to an external device.
- Connect the external device to the standalone URCVT computer.
- Refer to the documentation regarding installation, storage, and verification from the printer manufacturers such as HP, Canon, or any other printer manufacturer.

Verifying UPS installation:

- Verification of the UPS installation requires that the URCVT computer is connected to the correct UPS power source. No other drivers are required.
- As with all hardware, the equipment should be stored according to manufacturer recommendations with regard to temperature, humidity, and other external factors.

Document Revision History

Date	Version	Description	Author
09/08/2021	1.1.1	Updated for general audience	Melissa Hall
05/09/2021	1.1.0	Updated list of software for URCVT and added processes for procuring verified versions of COTS software	RCVRC
04/25/2021	1.0.0	System Hardening Procedures	Rosemary F. Blizzard

Appendix 5

Software Design and Specifications

RCTab v.1.2.0 - 02 - Software Design and Specifications v.1.2.0

Coding Standards and Style

We use *Google Checkstyle for Java* as our published, reviewed, and industry-accepted code style. For details see [Google_Java_Style_Guide.html](#) in the appendix.

Code development and review processes are described in the ***RCTab v.1.2.0 - 13 - Quality Assurance Plan v.1.1.0.***

Java 14

The tabulator is written in Java because it meets the VVSG requirements for software language selection. It is widely supported and popular in both industry and the open-source community for a wide variety of applications. It offers a mature and robust collection of third-party libraries. The Java Runtime Environment is standard on our target platforms which means a simple installation process. Our specific version of Java is OpenJDK 14.0.1 and it can be downloaded [here](#).

Open Source

We develop the tabulator as an open-source project for three main reasons:

- 1) Transparency: published source code increases public confidence in the application by giving anyone the opportunity to review our work and the processes and methodology behind it.
- 2) Adoption: open-source licensing encourages others to use the software to facilitate the spread of ranked-choice voting.
- 3) Collaboration: open-source licensing enables other software developers to contribute enhancements to the project and incorporate it into other related projects (RCV visualizers, policy research, etc.)

Architecture

RCTab consists of one in-house java code module built from 23 source files. These are described in more detail under **Tabulator Java Classes** below. The Tabulator relies on basic java platform libraries (file I/O, string processing, logging) and several 3rd-party modules listed below for reading and writing various file formats. These code modules are compiled and packaged with a minimal java runtime environment (14.0.1) which executes compiled object code, when installed and run on the target system.

Tabulator Java classes

The following Java classes comprise the entirety of all in-house developed software and implement all core functionality of the Tabulator.

CastVoteRecord: The in-memory representation of each cast vote record read from a source file. When source files are first processed at the beginning of a tabulation, each `CastVoteRecord` object contains the data parsed from the source, including the candidate rankings, the precinct ID, and other relevant metadata. As the tabulation progresses, the object keeps track of the cast vote record's fate, including which candidate(s) this ballot is counting toward and whether it has been exhausted.

ClearBallotCvrReader: Contains the logic for parsing a CVR source file in Clear Ballot's comma-separated value format and populating a list of `CastVoteRecord` objects.

CommonDataFormatReader: Contains the logic for parsing a CVR source file in the Common Data Format and populating a list of `CastVoteRecord` objects. It supports both XML and JSON.

ContestConfig: A wrapper around `RawContestConfig`. It contains extensive validation to confirm that a config file contains parameters that are legal and consistent with one another. It also has logic for reading a config file from disk, preprocessing the candidate data from a config, and normalizing some of the config values for use during tabulation.

ContestConfigMigration: Provides support for identifying whether a config file's version is compatible with the version of the application that is running. It also contains logic for automatically migrating a config from an older version to make it compatible with the current version.

DominionCvrReader: Contains the logic for parsing a set of CVR source files in Dominion's JSON format and populating a list of `CastVoteRecord` objects.

FileUtils: A few simple utility functions for reading from and writing to directories on disk.

GuiApplication: The simple logic for launching the application's graphical user interface, including loading the main layout markup stored in the `GuiConfigLayout.fxml` file.

GuiConfigController: Contains most of the logic for the interactive components of the graphical user interface, ensuring that the application responds appropriately when the user clicks a button, menu item, or other interactive element.

GuiContext: A singleton class that supports the graphical user interface in managing file chooser dialogs for opening and saving config files.

GuiTiebreakerController: Supports the interactive logic for selecting a tie-breaker winner or loser in the graphical user interface when the tie-break mode is set to one of the interactive options.

HartCvrReader: Contains the logic for parsing a CVR source file in the Hart XML format and populating a list of `CastVoteRecord` objects.

JsonParser: Generic logic for reading JSON files from disk and writing them to disk. It's used by the JSON parsing code for parsing Common Data Format and Dominion CVR files, as well as for reading and writing tabulator config files.

Logger: Handles the formatting and saving to disk of audit and operator log files.

Main: The main entry point for the application. Depending on the arguments supplied, it either launches the graphical user interface or proceeds with a command-line-based tabulation.

RawContestConfig: A RawContestConfig object is a simple in-memory representation of a config file loaded from disk.

ResultsWriter: After a tabulation completes, a ResultsWriter generates all of the appropriate summary results files and saves them to disk. Results files can include a summary CSV spreadsheet file, a summary JSON file, and a full Common Data Format JSON file. When "tabulate by precinct" is enabled, it also produces separate summary files for each precinct.

StreamingCvrReader: Contains the logic for parsing a CVR source file in ES&S's XLS format and populating a list of CastVoteRecord objects. It also extracts a list of precinct IDs if tabulation by precinct is enabled.

Tabulator: The core logic for tabulating a contest given a list of CastVoteRecord objects and a ContestConfig, it runs the round-by-round tabulation, writing to the tabulation log file as it proceeds, and then calls ResultsWriter to generate results files when it completes.

TabulatorSession: Manages the process of running a single tabulation. Given the path to a config file, it loads and validates the config, loads and parses the cast vote record source files, and runs the tabulation, including generating the results files at the end. It also contains logic for converting an input config into a Common Data Format source file without actually running a tabulation.

TabulatorTests: Runs all of the regression tests. Each test involves loading a config file and, if it's valid, running the tabulation and then comparing the output summary JSON file to an existing file containing the expected output. If the config file has generateCdfJson enabled, it also compares the generated CDF JSON file to an existing file containing the expected version of this output.

TallyTransfers: The Tabulator class maintains a TallyTransfers object (and one per precinct if tabulating by precinct is enabled) to keep track of the number of votes transferring from each source to each destination in each round as candidates are eliminated or elected. This data is included in the results summary JSON to enable Sankey plot visualizations.

Tiebreak: Contains the logic for breaking a tie when the tabulation needs to select a candidate for elimination or election and multiple candidates are tied with the same current vote total.

Utils: Miscellaneous utility functions for processing strings and identifying the user's environment.

3rd-party Modules

RCTab incorporates several 3rd-party modules which are all open-source. These meet the VVSG requirements for third-party modules. They are mature and widely accepted and used. None of them are modified in any way.

Module Name	Version	Purpose	Link
Apache Commons CSV	1.8	CSV "Comma Separated Values" reader / writer.	https://commons.apache.org/proper/commons-csv/user-guide.html
Apache POI OOXML	4.1.2	Excel spreadsheet reader / writer.	https://poi.apache.org/apidocs/dev/org/apache/poi/ooxml/package-summary.html
Jackson Core	2.11.1	XML / JSON streaming reader / writer core	https://github.com/FasterXML/jackson-core
Jackson Annotations	2.11.1	XML / JSON deserialization annotations	https://github.com/FasterXML/jackson-annotations
Jackson Databind	2.11.1	XML / JSON deserialization	https://github.com/FasterXML/jackson-databind
Jackson Dataformat XML	2.11.1	XML reader / writer	https://github.com/FasterXML/jackson-dataformat-xml
Jupiter JUnit API	5.6.2	Automated testing (used only during development)	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
Jupiter JUnit Engine	5.6.2	Automated testing (used only during development)	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine

Software Limits:

Limitations of the Tabulation software i.e., how many CVRs can be tabulated are detailed in the ***RCTab v.1.2.0 - 03 - System Hardware Specification v.1.1.0*** document.

Contest Tabulation Logic

Overview

When the user triggers a contest tabulation, the application creates a new Tabulator Session object to manage the process flow. The Tabulator Session loads, parses, and validates the contest config file. If those steps succeed, it reads the cast vote records into memory, runs the tabulation, and generates the results summary files. Throughout these processes logging output is written to two locations as detailed below under **RCTab Logging**

- 1) Read the config file
- 2) Validate the config file
- 3) Read cvr files
- 4) Round by round tabulation of votes according to configuration
- 5) Generate reports

For detailed user guides, see ***RCTab v.1.2.0 - 08 - System Operations Procedures v.1.1.0*** and ***RCTab v.1.2.0 - 300 - Configuration File Parameters v.1.1.0***. Note that this document covers all tabulation logic in RCTab in order to fully describe the software design. All tabulation logic was defined according to the documents listed on page 12, in conjunction with the draft VVSG 2.0 1500-107 Voting Methods and Tabulation Models document (for which RCTab acts as a reference implementation), as well as with reference to all RCV laws currently in use in the United States. Contact vendor for a folder with all such RCV laws and/or a copy of that unpublished draft standard.

Reading a config file

When a user selects an existing config file to open, the tabulator parses the JSON file and stores the information in a RawContestConfig object. This object is wrapped inside a ContestConfig object. The ContestConfig object serves as an interface between the config file and the rest of the application, providing extensive validation logic and a number of convenience methods for accessing normalized versions of the config settings.

A config file includes a tabulatorVersion string and a collection of parameters organized into four sections.

The `tabulatorVersion` is used to confirm that the version of the application that's loading the config file is compatible (specifically: not older than) the version of the application that created the file.

The `outputSettings` section contains settings related to what forms of output should be generated and where it should be saved on disk.

The `cvrFileSources` section is a list of one or more source file paths containing cast vote records, along with the parameters necessary for the tabulator to parse the records successfully.

The `candidates` section lists all of the candidate names/codes that appear in any of the CVR source files.

The `rules` section contains all of the parameters that determine exactly how the application should tabulate the cast vote records and produce results.

Writing the config file

A user can create a new config file or open an existing one and edit it. The GUI allows the user to modify all of the parameters that populate a config file (except for `tabulatorVersion`, which is determined by the app), run a validation to confirm that all of the settings in the file are valid, and save the file.

Validation

Config file validation checks each aspect of the file and attempts to verify that all of the individual settings are compatible with one another and should result in a successful tabulation. This includes confirming that the tabulation rules are consistent, that the candidate names are valid and don't contain duplicates, and that each source cvr file exists on disk, and has the proper parameters set for that cvr provider. The GUI prevents the user from creating a config file that would fail many of these checks, but the validation assumes nothing and provides an additional layer of protection against user error. Because a user can create or modify a config file simply using a text editor the tabulator can't assume that a config file has only been modified by its own GUI.

Reading CVR Files

Reading the cast vote records into memory consists of parsing the source files and creating a `CastVoteRecord` object to represent each record. This object contains the candidate rankings

for the CVR along with additional information parsed from the source file such as an ID and a precinct name. As the tabulation progresses, the CastVoteRecord object also stores information about the CVR's fate in each round (which candidate(s) its vote is counting toward and what fraction of the vote belongs to each). The details of parsing each source file depend on that file's provider.

Tabulation of CVRs

Note: for all winning modes other than MULTI_SEAT_SEQUENTIAL_WINNER_TAKES_ALL (a.k.a. multi-pass IRV), the application performs a single tabulation when it processes a config file. For multi-pass IRV, it instead performs a sequence of single-seat tabulations in which each tabulation excludes the candidates who have won on prior iterations, continuing until it has run numWinners tabulations and identified numWinners winners.

For the tabulation itself, the application creates a Tabulator object. This object's tabulate method runs a loop that iterates until it determines that the tabulation is complete. Each iteration of the loop is a new round. Each round starts by calling computeTalliesForRound, which iterates over all of the cast vote records and sums up the total number of current votes for each candidate.

This involves a number of steps:

1. If the CVR has already been marked as exhausted, it is skipped.
2. If the CVR was counted toward a candidate in the previous round and that candidate is still active, it's counted toward that candidate again in this round.
3. If the CVR has no rankings at all, it is marked as exhausted.
4. The tabulator then begins iterating through the rankings in the CVR, starting with the most preferred rank found (i.e., the lowest rank number, which is typically 1) and proceeding in order. At each rank, it checks for a number of possible cases:
 - i. If the number of rankings skipped between the last ranking seen and this one exceeds the maxSkippedRanksAllowed value in the config, the CVR is marked as exhausted.
 - ii. If one or more of the candidates at this rank already appeared at another rank and the config has enabled exhaustOnDuplicateCandidate, the CVR is marked as exhausted.
 - iii. If the rankings at this rank constitute an overvote, the CVR is handled according to the overvote rule set in the config.
 - iv. If a continuing candidate is found at this rank, the CVR is marked as counting toward the candidate.
 - v. Otherwise, the CVR is marked as exhausted.
5. If the config has enabled tabulation by precinct, the method also updates the per-precinct tallies for this round.

Next, if the tabulation is in the first round and/or the contest is for a single seat or the mode is MULTI_SEAT_SEQUENTIAL_WINNER_TAKES_ALL (multi-pass IRV), the software sets/updates the winning threshold (the number of votes that a candidate must meet or exceed in order to be named a winner) based on the winning rules set in the config.

1. The threshold in single-seat and MULTI_SEAT_SEQUENTIAL_WINNER_TAKES_ALL is calculated in each round as:

$$\text{winningThreshold} = \text{floor}(V/2) + 1$$

Where V = the total number of votes counting for continuing candidates in the current round (and continuing refers to non-excluded candidates who have not yet been eliminated -- and, in the case of multi-pass IRV, who have not been elected in a previous pass)..

2. The threshold in MULTI_SEAT_ALLOW_ONLY_ONE_WINNER_PER_ROUND and MULTI_SEAT_ALLOW_MULTIPLE_WINNERS_PER_ROUND is calculated as:

$$\text{winningThreshold} = (V/(N+1)) + 10^{(-1 * \text{decimalPlacesForVoteArithmetic})}$$

nonIntegerWinningThreshold is set to true

$$\text{winningThreshold} = \text{floor}(V/(N+1)) + 1 \text{ if nonIntegerWinningThreshold is set to false}$$

$$\text{winningThreshold} = \text{floor}(V/N) + 1 \text{ if hareQuota is set to true}$$

Where V = the sum of the values in currentRoundCandidateToTally in the first round
and n = numWinners

3. The election threshold in MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD is calculated as:

$$\text{winningThreshold} = V * T$$

Where V = total number of votes counting for continuing candidates (candidates not eliminated) in the current round
and T = bottomsUpPercentageThreshold

No threshold is calculated in MULTI_SEAT_BOTTOMS_UP_UNTIL_N_WINNERS mode because the tabulation simply eliminates candidates until exactly numWinners remain, then selects those candidates as the winners.

The software then checks whether there are any continuing candidates with vote tallies in the current round that meet or exceed the winning threshold. Each of these candidates is marked as a winner and, if the winning rules in the config indicate that surplus votes should be redistributed (which is the case when the winning mode is either `MULTI_SEAT_ALLOW_ONLY_ONE_WINNER_PER_ROUND` and `MULTI_SEAT_ALLOW_MULTIPLE_WINNERS_PER_ROUND`), then the software calculates a surplus fraction. The surplus fraction for a winning candidate is calculated as:

$$\text{surplusFraction} = (C - T) / C$$

Where C = the candidate's vote tally in the current round
and T = the winning threshold

Each `CastVoteRecord` object counting towards a winning candidate redistributes an amount equal to its current value multiplied by the `surplusFraction` according to step 4 in "Tabulation of CVRs" above, based on that `CastVoteRecord`'s rankings. Each of these `CastVoteRecords` is also updated to record the portion of that vote that should remain allocated to the candidate in future rounds, which is its current value multiplied by $(1 - \text{surplusFraction})$.

Note that if a `CastVoteRecord` is involved in multiple surplus redistributions, the fraction of the vote allocated to earlier winners is not affected by subsequent redistributions; only the surplus portion is eligible for further redistribution.

Any decimal values in the above calculations are governed by the `decimalPlacesForVoteArithmetic` setting.

More information about how surplus vote values are calculated is included in the fractional transfers discussion in section 8 of ***RCTab v1.2.0 - 110 - Tabulation Options for RCV Tabulation v1.1.0***.

If the mode is `MULTI_SEAT_ALLOW_ONLY_ONE_WINNER_PER_ROUND` and more than one continuing candidate meets or exceeds the winning threshold in a round, only the candidate with the top tally is selected as the winner (and any other continuing candidates meeting the threshold will be selected in subsequent rounds). If multiple candidates are tied for this top tally value, the tie is broken according to the selected `tiebreakMode`.

In `MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD`, winners are selected in a given round only if every continuing candidate meets or exceeds the winning threshold.

Finally, if no winners have been identified in the current round, the software determines whether it should identify one or more candidates to eliminate. (This is true if a) the number of identified

winners is fewer than the number of seats for the contest, b) there are more than two candidates remaining and the winning rules specify a single-seat contest that should continue until only two candidates remain, or c) the winning mode is MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD.) In this case, the candidate(s) to be eliminated are identified by trying the following methods in order until one of them returns one or more candidates:

1. If the cast vote records include undeclared write-ins and this set of candidates have not been eliminated yet, select them.
2. If the config specifies a minimum vote threshold for a viable candidate and one or more continuing candidates has a tally below that threshold, select them.
3. If the config enables batch elimination, attempt to identify two or more candidates who can be eliminated via this process.
4. Otherwise, select the remaining candidate with the lowest tally. (If multiple candidates are tied for the lowest tally, select one according to the tie-breaking rule specified in the config.)

The tabulator then determines whether it should continue to the next round and repeat the process. It continues if one of the following conditions is true:

- a. For a single-seat contest, one of these is true:
 - i. A winner has not been identified yet.
 - ii. "Continue until two candidates remain" is enabled and one of these is true:
 1. There are more than two candidates remaining.
 2. The eliminations that reduced the number of remaining candidates to two happened in the current round. (This condition is included to ensure that the tabulator will generate an additional round showing the final vote tallies when only the last two candidates remain.)
- b. For a multi-seat contest, one of these is true:
 - i. The winning mode is MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD and no winners have been identified yet.
 - ii. The winning mode is not MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD and the number of winners identified is fewer than the number of seats.
 - iii. The winning mode is neither MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD nor MULTI_SEAT_BOTTOMS_UP_UNTIL_N_WINNERS and the number of winners equals the number of seats, but the final winners were identified in the current round. (Similar to above, this condition ensures that the tabulator will generate an additional round showing the final surplus redistribution in MULTI_SEAT_ALLOW_ONLY_ONE_WINNER_PER_ROUND and MULTI_SEAT_ALLOW_MULTIPLE_WINNERS_PER_ROUND.)

When the tabulator determines that it should not continue tabulating, the tabulation is complete.

Reporting results

After the tabulation completes, the software generates results files and saves them to disk. These results are based on the following data that the tabulation process has produced and stored in memory:

- Which round each eliminated candidate was eliminated
- Which round each winning candidate was identified as a winner
- The winning threshold that was used to select the winner(s)
- Each candidate's vote tally in each round
- If the config has enabled tabulating by precinct, each candidate's vote tally in each precinct in each round
- A record of the number of votes in each round that were transferred from each candidate to each other candidate (or were exhausted)
- In a multi-seat contest involving surplus redistribution, the cumulative amount of residual surplus in each round
- The number of exhausted ballots in each round, which are the number of ballots that cannot be counted for any continuing candidates - those candidates who are still active in the contest. This ranked-choice voting specific category of ballots includes undervoted ballots and overvoted ballots. Exhausted ballots are referred to as inactive ballots in summary results files.

Using this data, the tabulator creates a ResultsWriter object that writes two files to disk:

1. A CSV spreadsheet that includes the round-by-round tally for each candidate, when each candidate won or was eliminated, and related information
2. A JSON file that includes the same information found in the CSV spreadsheet, plus the number of votes transferred from each candidate to each other candidate (or exhaustion) in each round

If tabulating by precinct is enabled, the ResultsWriter also generates a CSV spreadsheet with round-by-round tallies for each precinct found in the cast vote records.

Finally, if the config file specifies that the tabulator should generate CDF (Common Data Format) output, it saves a CDF file in JSON format.

Note: if the winning mode is MULTI_SEAT_SEQUENTIAL_WINNER_TAKES_ALL (multi-pass IRV), the output files described in this section are generated after each pass, i.e., after each single-seat tabulation. The pass number is appended to each filename to allow the user to distinguish among them.

RCTab Logging

In addition to results files, RCTab generates various log outputs which describe:

- Program status
- Operations in progress
- Critical errors and warnings
- Tabulation-specific data: e.g., config file contents and how each vote counted in each round
- Additional system information

Log output is logically divided into two data streams:

Stream 1: Tabulator "Operator" Logging

This data stream captures the overall operation of the Tabulator which may include loading, editing, validating, and saving multiple config files, running cvr conversion functions, and tabulating multiple contests.

File Name: rcv_0.log

File Rotation:

When rcv_0.log reaches 50MB size it will be renamed along with any other preceding log files.

For example:

rcv_0.log -> rcv_1.log

rcv_1.log -> rcv_2.log

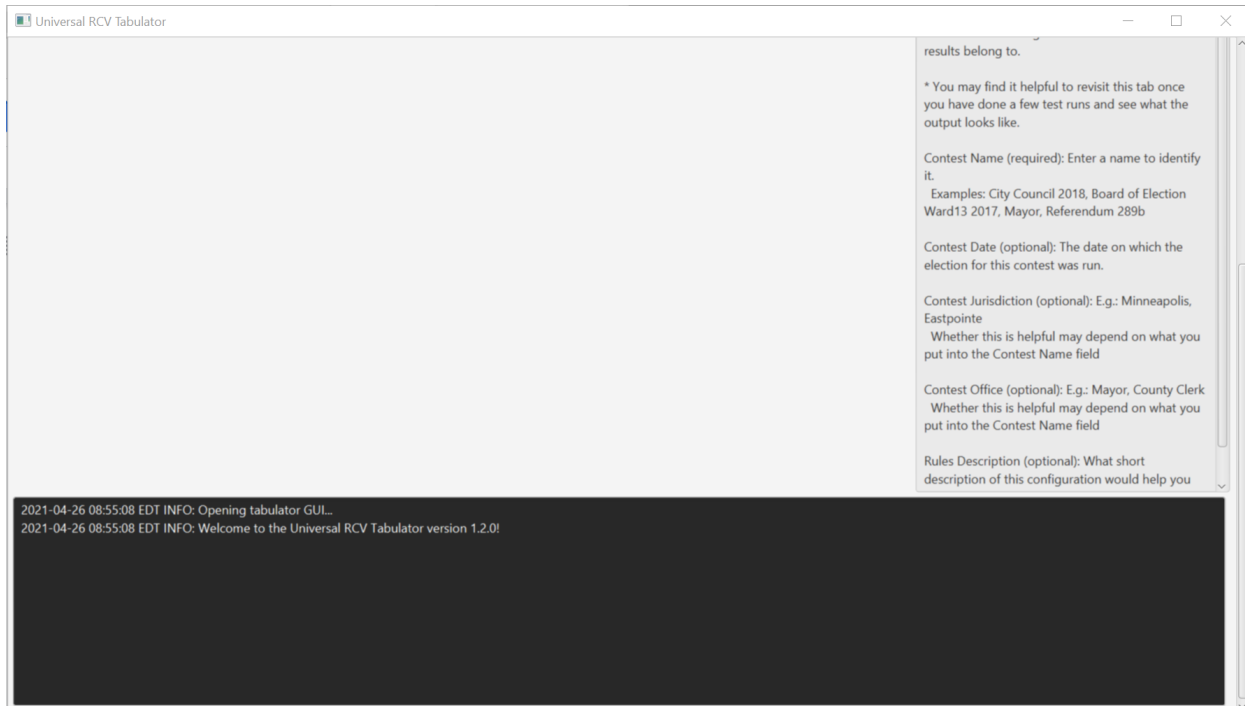
Then a new rcv_0.log file will be created, and logging will continue.

This is a standard log rotation strategy that limits log file sizes for easier management.

File location: The Tabulator Operator Log will be created in the current working directory. When launched using the rcv.bat file this is next to the rcv.bat file. For example:

C:\Users\MyUser\rcv\bin\rcv_0.log

Operator logging is duplicated in the black GUI console box at the bottom of the application window. The console is provided as a convenience primarily for showing validation errors and providing feedback to the user. See screenshot below for example of this console box.



Stream 2: Contest-Specific "Audit" Logging

This data stream captures information about a specific contest tabulation. It includes all the same information written to the Execution Log (within the context of a single contest tabulation) and includes a listing of the config file being tabulated, and a record of how each cvr was counted in each round.

This data stream only begins logging after a config file has been validated. Thus, all config validation logging (including any validation failures) will only appear in the execution log. We recognize that this behavior is non-intuitive, and we have filed an issue to improve it:

<https://github.com/BrightSpots/rcv/issues/125>

File Name: [time_stamp]_audit_0.log where timestamp is created when the tabulation is triggered and used on all output files for a given tabulation. For example:

2021-04-24_22-49-49_audit_0.log

File Rotation uses the same strategy as the Execution Logging. When an audit log reaches 50MB size it will be renamed along with any other preceding log files. For example:

2021-04-24_22-49-49_audit_0.log -> 2021-04-24_22-49-49_audit_1.log

2021-04-24_22-49-49_audit_1.log -> 2021-04-24_22-49-49_audit_2.log

Then a new 2021-04-24_22-49-49_audit_0.log file will be created, and logging will continue.

File Folder Location is specified in the config file under "outputDirectory"

User Interface:

The tabulator was originally developed for a command-line interface. The GUI (graphical user interface) was introduced both to make the process of configuring and running a tabulation faster and more intuitive and to enable users with a less technical background to use the software. The command-line interface still exists as a way to support execution via script, e.g., for batch processing and test automation. The GUI prevents the user from creating a config file that would fail many of these checks. A brief user guide to the Command Line Interface is available in ***RCTab v.1.2.0 - 240 - Command Line Instructions v.1.1.0***. More information about error messages can be found in ***RCTab v.1.2.0 - 430 - RCTabOperator Log Messages v.1.1.0***.

Supported File types:

The Tabulator uses the JSON format for contest configuration files and one style of results summary output. JSON is simple, popular, and easy for humans and software to read and write. The Tabulator uses CSV (comma-separated values) for the tabular version of its results summary output. CSV is a non-proprietary format that all modern spreadsheet applications can recognize. Tabulation output log files are produced in plain-text with a .log extension for clarity. RCTab reads .xlsx (Microsoft Excel) and .xml (Extensible Markup Language) cvr and elections metadata files supplied by various vendors (ES&S, Hart, CDF, Unisyn).

Additionally, the following documents were used to design the RCTab software:

RCTab v.1.2.0 - 110 - Tabulation Options for RCV Tabulation v.1.1.0
RCTab v.1.2.0 - 100 - Ranked-Choice Voting Laws v.1.1.0
RCTab v.1.2.0 - 120 - Process Ranked Choice Voting Contest v.1.1.0
RCTab v.1.2.0 - 130 - Expected Outcome RCV Test Sets Single-Winner v.1.1.0
RCTab v.1.2.0 - 140 - Expected Outcome RCV Test Sets Multi-Winner v.1.1.0
RCTab v.1.2.0 - 150 - ES&S Ballot Limitations & Maximum Testing Range v.1.1.0
RCTab v.1.2.0 - 07 - System Security Specification Requirements v.1.1.0

V.2:2.5.2 Applicable Documents

The vendor shall list all documents controlling the development of the software and its specifications. Documents shall be listed in order of precedence.

- This is described in this doc under **Coding Standards and Style**

V.2:2.5.3 Software Overview

The vendor shall provide an overview of the software that includes the following items:

c. Identification of all software items, indicating items that were:

1) Written in-house.

- All in-house items written in-house are described in this doc under **Summary of Java Classes**

2) Procured and not modified; and

- All third-party modules are not modified. They are described in this doc under **Java 14, Architecture, 3rd-party Modules**

3) Procured and modified including descriptions of the modifications to the software and to the default configuration options. The vendor shall also include a certification that procured software items were obtained directly from the manufacturer or a licensed dealer or distributor.

- No third-party modules have been modified.

We affirm that all third-party modules were obtained from the manufacturer. All of the third-party modules can be freely obtained and updated via the internet. They are maintained by their respective authors and hosted in the Maven Central Repository.

V.2:2.5.4 Software Standards and Conventions

The vendor shall provide information that can be used by an accredited test lab or state certification board to support software analysis and test design. The information shall address standards and conventions developed internally by the vendor as well as published industry standards that have been applied by the vendor. The vendor shall provide information that addresses the following standards and conventions:

a. Software System development methodology.

Our software development methodology is feature-driven:

- New customer needs are identified and specified
- Features are designated for a particular release or deferred
- Design new or updated software to meet those needs
- Software is created or updated, reviewed and tested (see 13 - Quality Assurance Plan)
- Software is submitted for VSTL testing
- Iterate with test lab
- Release

b. Software design standards, including internal vendor procedures.

c. Software specification standards, including internal vendor procedures.

d. Software coding standards, including internal vendor procedures.

e. Testing and verification standards, including internal vendor procedures, that can assist in determining the program's correctness and ACCEPT/REJECT criteria; and
f. Quality assurance standards or other documents that can be used by the ITA to examine and test the software. These documents include standards for program flow and control charts, program documentation, test planning, and for test data acquisition and reporting.

- Our software design and coding standards are based on the VVSG 1.0 VOL 1: 5.2 Software Design and Coding Standards. Additionally, we use Google Checkstyle as described in this document under **Coding Standards and Style**.
- Software specifications are described in our Github repo under the related issues. Our software specification standard is ad-hoc.
- Testing and verification standards are described in **RCTab v.1.2.0 13 Quality Assurance Plan v.1.1.0** and **RCTab v.1.2.0 17 System Test & Verification v.1.1.0**.

V.2:2.5.5 Software Operating Environment

This section shall describe or make reference to all operating environment factors that influence the software design.

- RCTab is designed to run on unmodified COTS Windows, Mac, and Linux operating systems as they are very popular, robust, mature, trusted, and they all support Java 14. For more information see the above section titled **Java 14**.
- RCTab core processing and tabulation logic is entirely platform (i.e., operating system) agnostic.
- All platform-specific functionality is encapsulated in the Java language implementation and Java platform libraries, including:
 - Basic OS services for disk IO
 - File path resolution
 - Memory management
 - System interrupt handling.

V.2:2.5.5.1 Hardware Environment and Constraints

The vendor shall identify and describe the hardware characteristics that influence the design of the software, such as:

- a. The logic and arithmetic capability of the processor.
- b. Memory read-write characteristics.
- c. External memory device characteristics.
- d. Peripheral device interface hardware.
- e. Data input/output device protocols; and
- f. Operator controls, indicators, and displays.

- RCTab is designed to be as efficient as possible with RAM and CPU usage in order to lower the hardware requirements needed to process an election and decrease costs for our users.
- Because the Tabulator creates an in-memory record for each CVR it processes, the maximum number of CVRs that can be processed in a single contest is limited by the amount of available RAM on the host system. See **RCTab v.1.2.0 - 03 - System**

Hardware Specification v.1.1.0 for more details.

V.2:2.5.5.2 Software Environment

The vendor shall identify the compilers or assemblers used in the generation of executable code and describe the operating system or system monitor.

- We use the javac compiler javac 14.0.1 included with OpenJDK 14.0.1 (build 14.0.1+7).
- See **RCTab v.1.2.0 14 Tabulator Build & Hashing Instructions v.1.1.0** for more details.

V.2:2.5.6 Software Functional Specification

The vendor shall provide a description of the operating modes of the system and of software capabilities to perform specific functions.

For more information about operating modes of the system and of system capabilities to perform specific functions, please see:

- **RCTab v.1.2.0 - 01 - System Overview v.1.1.0**
- **RCTab v.1.2.0 - 18 - User Guide v.1.1.0**
- **RCTab v.1.2.0 - 110 - Tabulation Options for RCV Tabulation v.1.1.0**
- **RCTab v.1.2.0 - 08 - System Operations Procedures v.1.1.0**
- **RCTab v.1.2.0 - 300 - Configuration File Parameters v.1.1.0**

V.2:2.5.6.1 Configurations and Operating Modes

The vendor shall describe all software configurations and operating modes of the system, such as ballot preparation, election programming, preparation for opening the polling place, recording votes and/or counting ballots, closing the polling place, and generating reports. For each software function or operating mode, the vendor shall provide:

- a. A definition of the inputs to the function or mode (with characteristics, tolerances or acceptable ranges, as applicable);
- b. An explanation of how the inputs are processed; and
- c. A definition of the outputs produced (again, with characteristics, tolerances, or acceptable ranges as applicable).

For more information about the components of RCTab with regard to data input/counting and generating final count that are relevant here, please see:

- **RCTab v.1.2.0 - 01 - System Overview v.1.1.0**
- **RCTab v.1.2.0 - 18 - User Guide v.1.1.0**
- **RCTab v.1.2.0 - 110 - Tabulation Options for RCV Tabulation v.1.1.0**
- **RCTab v.1.2.0 - 08 - System Operations Procedures v.1.1.0**
- **RCTab v.1.2.0 - 300 - Configuration File Parameters v.1.1.0**

V.2:2.5.6.2 Software Functions

The vendor shall describe the software's capabilities or methods for detecting or handling:

- a. Exception conditions.
- b. System failures.
- c. Data input/output errors.
- d. Error logging for audit record generation.

- Exceptions are handled using Java language built-in exception handling.
- Exceptions are logged to the **Operator Log** and, if a contest is being tabulated, the **Audit Log**.
- Tabulator Exceptions are detailed in **RCTab v.1.2.0 - 430 - RCTab Tabulator Operator Log Messages 1.0.0**.
- System failures such as out of memory, disk input/output errors, etc.. will be logged in these locations as well.

- e. Production of statistical ballot data.

- Tabulator results are described in **RCTab v.1.2.0 - 18 - User Guide v.1.1.0** under ***Running A Tabulation***.

- f. Data quality assessment; and

- For data quality assessment, we only operate upon CVR data formatted according to the standards the vendors have defined for their CVRs. See election system vendor documentation for more information regarding CVR formatting.
- Additionally, audit logs describe how all ballot data and configuration data in a contest tabulation was read/interpreted by RCTab.

- g. Security monitoring and control.

- Jurisdiction practices and policies govern the security monitoring control when using the tabulator. We recommend a minimum of 2 bipartisan, trained employees operate the system together at all times.
- For more information about security related issues, please see also **RCTab v.1.2.0 - 07 - System Security Specification Requirements v.1.1.0**.
- If there are additional questions outside the scope of the jurisdiction practices and policies or the documentation provided with the software regarding security issues, we recommend consulting with an information technology security specialist.

V.2:2.5.7 Programming Specifications

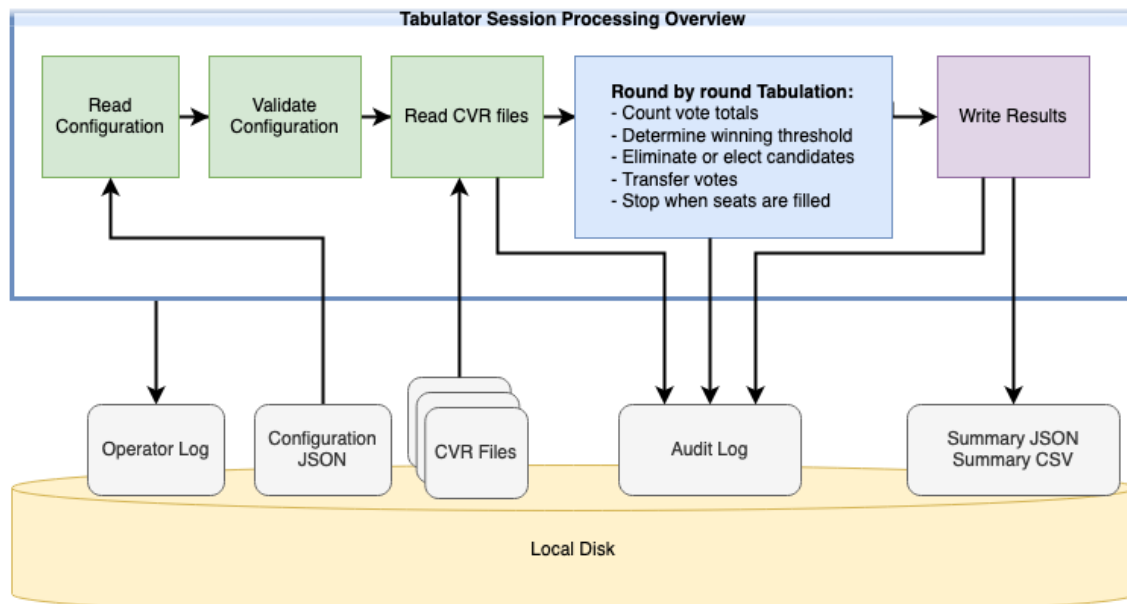
The vendor shall provide in this section an overview of the software design, its structure, and implementation algorithms and detailed specifications for individual software modules.

This is described in this document under ***Contest Tabulation Logic***. Also see the below diagram located in v.2:2.5.7.1.

V.2:2.5.7.1 Programming Specifications Overview

This overview shall include such items as flowcharts, HIPOs, data flow diagrams, and other graphical techniques that facilitate understanding of the programming specifications. This section shall be prepared to facilitate understanding of the internal functioning of the individual software modules. Implementation of the functions shall be described in terms of the software architecture, algorithms, and data structures.

This is described in this document under **Contest Tabulation Logic**. Also see the below diagram:



V.2:2.5.7.2 Programming Specifications Details

The programming specifications shall describe individual software modules and their component units, if applicable. For each module and unit, the vendor shall provide the following information:

- Module and unit design decisions, if any, such as algorithms used.
- Any constraints, limitations, or unusual features in the design of the software module or unit.
- If the software module or unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements as applicable.

- The programming specifications that describe individual software modules and their component units are described on pages 5-7 of this document.
- For additional information, see the following documents:
 - RCTab v.1.2.0 - 01 - System Overview v.1.1.0**

- **RCTab v.1.2.0 - 18 - User Guide v.1.1.0**
- **RCTab v.1.2.0 - 110 - Tabulation Options for RCV Tabulation v.1.1.0**
- **RCTab v.1.2.0 - 08 - System Operations Procedures v.1.1.0**
- **RCTab v.1.2.0 - 300 - Configuration File Parameters v.1.1.0**

f. If the software module or unit contains logic, the logic to be used by the software unit, including, as applicable:

- 1) Conditions in effect within the software module or unit when its execution is initiated.
- 2) Conditions under which control is passed to other software modules or units.
- 5) Exception and error handling:

The below list summarizes the basic function of all java classes, including conditions impacting software operation. Also see pages 2–3, 6-10 of this document.

V.2:2.5.8: Databases

The RCTab software does not use any databases for its operation. This requirement does not apply to RCTab.

V.2:2.5.9 Interfaces

RCTab does not rely on any interfaces for its operation. This requirement does not apply to RCTab.

Document Revision History

Date	Version	Description	Author
01/07/2022	1.2.0	Updated URCVT to RCTab and removed NY from the document.	Ryan Kirby
05/10/2021	1.1.0	Updated documentation for improved overall VVSG alignment	RCVRC
04/27/2021	1.0.0	Software Design Specifications	Louis Eisenberg

Appendix

The Appendix contains the following files:

1. TabulatorTests_example_console_output
2. Google JAVA Style Guide

Appendix 6

System Test and Verification Specification

RCTab

RCTab v.1.2.0 17- System Test and Verification Specification v.1.3.0

V.2:2.7 System Test and Verification Specification

- a. The vendor shall provide test and verification specifications for: Development test specifications.
- b. The vendor shall provide test and verification specifications for: National Certification test specifications.

V.2:2.7.1 Development Test Specifications

The vendor shall describe the plans, procedures, and data used during software development and system integration to verify system logic correctness, data quality, and security.

- a. This description shall include: Test identification and design, including:
 - i. Test structure
 - ii. Test sequence or progression; and
 - iii. Test conditions

Test Structure:

Included below on pages 8-10 is the basic template used to define every setting for each test set used to test the RCTab. This form is filled out with the settings for the relevant test. Testers can compare the information in this form to the information in the configuration file loaded from the relevant folder in test_data to ensure that all settings are correct. Each setting selected in this form has been tested. RCTab has not yet created a Test Structure Form for every RCTab test but will set up a form for all RCTab regression tests based on this form. Questions about this form can be made to the Ranked Choice Voting Resource Center at info@rcvresources.org or by calling 1-833-868-3728. We also have included an example Test Structure Form filled out according to the requirements of the Portland 2015 Mayor test. That example starts on page 11.

Regression Test Overview:

We have developed a suite of 57 Tabulation Regression Tests and continue to add more tests as new features and bug fixes are added. These are designed to verify that various aspects of Tabulator functionality behave as expected. They also verify that new code changes do not inadvertently alter Tabulator behavior. The entire test suite must be run, and all tests must pass before any new code changes can be merged into the main Tabulator repository.

Design Overview:

Each test contains a set of normal tabulation inputs (config file and cvr files) and a known valid "expected" results summary file. The test runs a tabulation with new code, and compares the results of that tabulation to previous, expected, correct results. In essence, we isolate and analyze the effects any new code changes may have on the tabulation output. This is a classic regression test design.

Test Execution Details:

The test suite will run through all tests automatically as follows:

1. Tabulator is built from source code.
2. For each test:
 - a. Run a tabulation using the test config file and cvr files.
 - b. Compare tabulation output summary file to reference expected summary file.
 - c. If the files match exactly (except for timestamps) the test passes.
 - d. If the files do not match the test fails.
 - e. Test execution and test results are written to a log file and console.

Testing Procedures:

When new code is ready for submission, a developer will follow these procedures to ensure the code is safe for incorporation:

1. Run test suite: in a console, from the rcv root directory, enter: `./gradlew.bat test`
2. Observe the test output. If *any* test fails the source of the failure must be identified and either:
 - a. Fixed. Usually, a test failure is caused by a bug in logic or data which can be fixed.
 - b. Update the reference test asset. Sometimes bug fixes cause tests to fail because they are now operating correctly. In these cases, test output must be manually verified and peer-reviewed (like any code change) before it can be updated.

Example test outputs are included in: `TabulatorTests_example_console_output.txt` and `Test_Results_TabulatorTests.html`

For actual test data, including configuration files and expected results, refer to `test_data.zip` that includes all test data and was submitted with documentation.

Users can also manually run each test. The steps required to run tests manually are as follows:

1. Open RCTab
2. Click "File"
3. Click "Load"

4. Navigate to the configuration file for the test you wish to run
5. Select the configuration file and load it into the RCTab
6. Confirm that the configuration file properly loaded into the RCTab by checking that the relevant fields are filled in
7. Click “Tabulation”
8. Click “Tabulate”
9. Wait for the RCTab to finish tabulating
10. Navigate to the output location for your results
11. Compare your results .json summary file to the .json summary file included with the data. If this matches exactly, the test passes.

Test Conditions:

Test conditions require the following files for input for each test case: Configuration file (JSON), Cast Vote Record (typically CSV format). Confirm appropriate files are uploaded to proceed with appropriate testing conditions.

For more information on the development process as it relates to testing see ***RCTab v.1.2.0 13-Quality Assurance Plan v.1.1.0***.

Additional Testing Procedures:

In addition to regression testing of all changes, new features are tested by developers and RCVRC staff to ensure that they work as expected. If any features did not work as expected, a ticket was filed on GitHub with the test data and an explanation of how the achieved results differed from the expected results.

The RCVRC and Bright Spots also conduct scale tests of the RCTab. Tests include a contest with 100 CVR files composed of 100,000 individual cast vote records each, a contest with 1,000 CVR files composed of 1,000,000 individual cast vote records each, a contest with 6,000 CVR files composed of 6,000,000 individual cast vote records each, and a set of 11 CVRs composed of 9,200,000 individual cast vote records each. Volume tests with these records were conducted throughout March and April 2021.

c. This description shall include: Special purpose test procedures including any assumptions or constraints;

- No special-purpose test procedures are followed.

d. This description shall include: Test data, test data source, whether it is real or simulated, and how test data are controlled;

Test Data:

The data for 2017 Minneapolis Mayor, 2013 Minneapolis Mayor, 2013 Minneapolis Park, 2018 Maine Governor Democratic Primary are all from real-world elections and were procured directly from the websites of the jurisdictions listed in the name of the data. All other data was manufactured by Bright Spots developers, RCVRC staff, voting system vendors, or election jurisdictions. Any data with vendor names included (Unisyn, Clear Ballot, Dominion, Hart, CDF) was procured from vendors directly or from jurisdictions working with those vendors. Any data not falling in those categories was manufactured by RCTab developers according to the CVR requirements of the software to test specific functionalities of the RCTab software.

Regression test data are controlled by hosting them on GitHub and updating tests as new features are added to the RCTab. Other test data, for tests run by RCVRC staff, Bright Spots developers, or others, did not previously have clear controls on data. Data was procured from trusted sources, such as actual election jurisdictions and voting system vendors, as much as possible or generated according to the specifications of previously used CVR data from vendors. However, no formal controls were in place for test data or for tracking results. Results were communicated via the relevant issue(s) on GitHub. We are implementing formal control methods for tests and test data going forward.

e. This description shall include: Expected test results;

- Expected results for regression tests are included in the test folders available in the `test_data.zip` that includes all test data and was submitted with documentation.

f. This description shall include: Criteria for evaluating test results.

The names of the tests are mostly self-explanatory. For example, `test_set_1_exhaust_at_overnote` tests whether the tabulator correctly interprets the `overnoteRule = "exhaust immediately"` setting. Some of the tests aren't testing specific features but instead are testing a known data set, e.g., `2018_maine_governor_primary`. The complete list is included below.

Results from any tests run on regression test data should 100% match the expected results in each individual test's folder on GitHub. Note that test folders include only .json summary result files, while tests run by users will produce .json summary results, .csv summary results, and .log audit log files. The .json summary results should therefore be compared when evaluating test results.

In tests run by RCVRC staff and Bright Spots developers when incorporating new features, any features were tested according to requirements as laid out in RCV laws or regulations being incorporated into the RCTab.

V.2:2.7.2 National Certification Test Specifications

The vendor shall provide specifications for verification and validation of overall software performance.

a. These specifications shall cover: Control and data input/output;

- Data input: Data used in the RCTab should come directly from the user jurisdiction certified voting system. When exporting the data from the certified voting system the users should adhere to all export procedures as outlined by the voting system vendor.
- Data output: Data created via the RCTab should be created using the guidelines and procedures outlined in **RCTab v.1.2.0 18- User Guide v.1.1.0**.

b. These specifications shall cover: Acceptance criteria;

- Tests in the set of regression tests are designed to test one or more functionalities of the RCTab. When each test is performed, the results of the test will reveal whether the RCTab meets the functionality described in the TDP. Information about how each functionality of the RCTab is intended to function is provided in **RCTab v1.2.0 02- Software Design and Specifications v.1.1.0** and **RCTab v.1.2.0 18- User Guide v.1.1.0**.

c. These specifications shall cover: Processing accuracy;

- All functions tested must produce a result that matches the expected results. Matching results means a test passed this requirement. Processing accuracy procedures are outlined in **RCTab v.1.2.0 18- User Guide v.1.1.0**.

d. These specifications shall cover: Data quality assessment and maintenance;

- Data quality is tested by ensuring that data used in the RCTab comes directly from the user jurisdiction certified voting system and by producing expected results for a test before running a test itself. Once a test is run, results should be inspected to confirm that they match the expected outcome. Data should be maintained on secure drives or secured computers/networks, as required by the user jurisdiction.

e. These specifications shall cover: Ballot interpretation logic;

- Regression tests can be used to test ballot interpretation logic. All CVR data will include discrete and clearly defined values for how each ranking on an RCV ballot was used. Ballot interpretation via the RCTab relies upon the candidate names, winning election rules, and voter error rules a user specifies in a configuration file. If the produced results match the expected results, then a test passed the ballot interpretation logic requirement.

f. These specifications shall cover: Exception handling;

- The `invalid_params_test` and `invalid_sources_test` can be used to test exception handling in the RCTab. When run through the RCTab these tests will cause the RCTab to produce errors in the operator log box at the bottom of the user interface. If those errors shown match those in the below screenshot, the test was successful.

```
2021-05-13 07:19:56 EDT INFO: Validating contest config...
2021-05-13 07:19:56 EDT SEVERE: contestName is required!
2021-05-13 07:19:56 EDT SEVERE: Contest config must contain at least 1 cast vote record file!
2021-05-13 07:19:56 EDT SEVERE: Contest config must contain at least 1 declared candidate!
2021-05-13 07:19:56 EDT SEVERE: Invalid tiebreakMode!
2021-05-13 07:19:56 EDT SEVERE: Invalid overvoteRule!
2021-05-13 07:19:56 EDT SEVERE: Invalid winnerElectionMode!
2021-05-13 07:19:56 EDT SEVERE: numberOfWinners must be an integer from 0 to 2147483647!
2021-05-13 07:19:56 EDT SEVERE: Contest config validation failed! Please modify the contest config file and try again.
```

g. These specifications shall cover: Security;

- RCTab has no tests specifically designed to test security. RCTab relies on a jurisdiction's security policies, as laid out in **RCTab v.1.2.0 06- System Security Specifications v.1.1.0**. Security can be maintained by ensuring that all security procedures follow the requirements set out in **RCTab v.1.2.0 06- System Security Specifications v.1.1.0**.

h. These specifications shall cover: Production of audit trails and statistical data.

- Regression tests will produce .log audit files and will log general information about the contest tabulation to the operator .log file. Production of these files can be tested using any regression test files, so long as the user activates the "Tabulate" function. All tests that produce a successful tabulation will also create .csv summary results files and .json summary results files. If these .log files are generated and updated, and .csv and .json summary files are produced when the RCTab successfully completes a tabulation, then this requirement passes.

The specifications shall identify procedures for assessing and demonstrating the suitability of the software for election use.

- For additional user jurisdiction testing specifications, please see **RCTab v.1.2.0 05- Acceptance Test Procedures v.1.1.0** and **RCTab v.1.2.0 11- L&A Testing v.1.1.0** to review vendor procedures to ensure the user jurisdiction has received and is using the correct trusted build for the state of .
- **RCTab v.1.2.0 18- User Guide v.1.1.0** also includes a detailed guide to expected user operation of the RCTab, which can be used to create tests of the user interface, tabulation functionalities, and other functionalities of the RCTab software. The vendor is available to support development of additional tests.

Document Revision History

Date	Version	Description	Author
01/05/2022	1.3.0	Updated document to reflect RCTab and remove NY	Rene Rojas
05/14/2021	1.2.0	Added test conditions. Updated Test data link to include file with all results. Added test structure information/form.	Kelly Sechrist
05/10/2021	1.1.0	Update to provide more details on past test behaviors and future plans for improving test tracking and control.	Chris Hughes
05/02/2021	1.0.0	System Test and Verification	Chris Hughes

RCTab Test Structure Form:

RCTab Test Structure Form

This form is the basic template used to define every test setting for each test set used to test the RCTab. This form is filled out with the settings for the [TEST NAME]. Testers can compare the information in this form to the information in the configuration file loaded from the relevant folder in test_data to ensure that all settings are correct. Each setting selected in this form is a setting tested by this test. RCTab has not yet created a Test Structure Form for every RCTab test, but will set up a form for all RCTab regression tests based on this form. Questions about this form can be made to the Ranked Choice Voting Resource Center at info@rcvresources.org or by calling 1-833-868-3728.

Contest Info			
Contest Name*:		Contest Date:	
Contest Jurisdiction:		Contest Office:	
Rules Description:			
CVR Files			
Provider*:		Enter Vendor (ES&S, etc.)	
File Path*:		Location of the cvr export file.	
First Vote Column Index*:		Enter the Column where the First Vote is in the CVR export file.	
First Vote Row Index*:		Enter the Row where the First Vote is in the CVR export file.	
ID Column Index:		Enter the ID Column (if being used)	
Precinct Column Index:		Enter the Precinct Column in the CVR export file.	
Overvote Label		ES&S will default to "overvote"	
Undervote Label		ES&S will default to "undervote"	
Undeclared Write-In Label		User will define. Must match case and spelling with CVR	

Treat Blank as Undeclared W1			Circle One
Candidates			
Name*:		<i>Obtain a list of candidates from the EMS System (Attach the list of candidates to this form)</i>	
Code:		<i>Enter the Code for each candidate (if being used) example: DDE</i>	
Excluded:		<i>Check box if a candidate is not being counted in this tabulation</i>	
Winning Rules			
Winner Election Mode*		Any user jurisdiction guidelines which identify the specific rules for ranked choice voting elections should be attached to this form for easy reference. Please see the RCTab User Guide for more information about the available selections that can be made.	
Maximum # of Ranked Candidates*			
Minimum Vote Threshold			
Use Batch Elimination	Y N (Circle One)		
Continue until Two Candidates Remain	Y N (Circle One)		
Tiebreak Mode*			
Random Seed*			
Number of Winners*			
Percentage Threshold*			
Threshold Calculation Method*			
Most Common Threshold			
HB Quota			
Hare Quota†			
Decimal Places for Vote Arith (MW ONLY)*			
Voter Error Rules			
Overvote Rule*		Choose one of the three available options.	
Skip to next rank			
Exhaust Immediately			
Exhaust if mult. cont.			

Consecutive Skip Ranks Allowed				Enter the number of consecutive skipped rankings or check Unlimited if permitted.	
Exhaust on multi ranks for same candidate		Y N (Circle One)		Check if multiple ranks for the same candidate are not permitted.	
Output					
Output Directory (where should results file go on PC?):					
Tabulate by Precinct:		Y	N (Circle One)	Generate CDF JSON:	Y N (Circle One)
Configuration File Name:					
Date of Test:				Passed?: YES NO	
Name of tester(s)					
Name 1		Name 2			

†Disclaimer: The Hare Quota tabulation option in the Universal RCV Tabulator software has not been thoroughly tested in a controlled testing lab environment. Do not attempt to implement this option without first testing in a non-operational environment. Please contact the Ranked Choice Voting Resource Center for additional information.

RCTab Test Structure Form Example:

RCTab Test Structure Form

This form is the basic template used to define every test setting for each test set used to test the RCTab. This form is filled out with the settings for the 2015 Portland Mayor test. Testers can compare the information in this form to the information in the configuration file loaded from the relevant folder in test_data to ensure that all settings are correct. Each setting selected in this form is a setting tested by this test. RCTab has not yet created a Test Structure Form for every RCTab test, but will set up a form for all RCTab regression tests based on this form. Questions about this form can be made to the Ranked Choice Voting Resource Center at info@rcvresources.org or by calling 1-833-868-3728.

Contest Info			
Contest Name*:	Portland 2015 Mayoral Race	Contest Date:	2015-11-03
Contest Jurisdiction:	Portland, ME	Contest Office:	Portland, ME
Rules Description:			
CVR Files			
Provider*:	ES&S	Enter Vendor (ES&S, etc.)	
File Path*:	2015_portland_mayor_cvr.xlsx	Location of the cvr export file.	
First Vote Column Index*:		4	Enter the Column where the First Vote is in the CVR export file.
First Vote Row Index*:		2	Enter the Row where the First Vote is in the CVR export file.
ID Column Index:	Leave blank		Enter the ID Column (if being used)
Precinct Column Index:		2	Enter the Precinct Column in the CVR export file.
Overvote Label	overvote	ES&S will default to "overvote"	
Undervote Label	undervote	ES&S will default to "undervote"	
Undeclared Write-In Label	UWI	User will define. Must match case and spelling with CVR	
Treat Blank as Undeclared WI	FALSE	Circle One	
Candidates			

Name*:	See below for a list of candidates.	Obtain a list of candidates from the EMS System (Attach the list of candidates to this form)
Code:		Enter the Code for each candidate (if being used) example: DDE
Excluded:		Check box if a candidate is not being counted in this tabulation
Winning Rules		
Winner Election Mode*	singleWinnerMajority	Any user jurisdiction guidelines which identify the specific rules for ranked choice voting elections should be attached to this form for easy reference. Please see the RCTab User Guide for more information about the available selections that can be made.
Maximum # of Ranked Candidates*	15	
Minimum Vote Threshold	0	
Use Batch Elimination	Y N (Circle One)	
Continue until Two Candidates Remain	Y N (Circle One)	
Tiebreak Mode*	useCandidateOrder	
Random Seed*	N/A	
Number of Winners*	1	
Percentage Threshold*	N/A	
Threshold Calculation Method*	N/A	
Most Common Threshold	N/A	Any user jurisdiction guidelines which identify the specific rules for ranked choice voting elections should be attached to this form for easy reference. Please see the RCTab User Guide for more information about the available selections that can be made.
HB Quota	N/A	
Hare Quota†	N/A	
Decimal Places for Vote Arith (MW ONLY)*	N/A	
Voter Error Rules		
Overvote Rule*	Exhaust Immediately	Choose one of the three available options.
Skip to next rank	-	
Exhaust Immediately	X	
Exhaust if mult. cont.	-	
Consecutive Skip Ranks Allowed	1	Enter the number of consecutive skipped rankings or check Unlimited if permitted.

Exhaust on multi ranks for same candidate		Y	N (Circle One)	Check if multiple ranks for the same candidate are not permitted.	
Output					
Output Directory (where should results file go on PC?): output					
Tabulate by Precinct:		Y	N (Circle One)	Generate CDF JSON:	Y N (Circle One)
Configuration File Name: 2015_portland_mayor_config.json					
Date of Test:		Passed?: YES NO			
Name of tester(s)					
Name 1		Name 2			

†Disclaimer: The Hare Quota tabulation option in the Universal RCV Tabulator software has not been thoroughly tested in a controlled testing lab environment. Do not attempt to implement this option without first testing in a non-operational environment. Please contact the Ranked Choice Voting Resource Center for additional information.

Candidates	Brennan, Michael F.					
	Bragdon, Charles E.					
	Bryant, Peter G.					
	Carmona, Ralph C.					
	Dodge, Richard A.					
	Duson, Jill C.					
	Eder, John M.					
	Haadoow, Hamza A.					
	Lapchick, Jodie L.					
	Marshall, David A.					
	Mavodones, Nicholas M. Jr.					
	Miller, Markos S.					
	Rathband, Jed					
	Strimling, Ethan K.					

	Vail, Christopher L.							
--	----------------------	--	--	--	--	--	--	--

List of Tests

Name of Test	Name of Test Folder	Description of test	Purpose of Test
RCVRC & Bright Spots name for regression test	<u>Test folder name in:</u> Refer to test_data.zip that includes all test data and was submitted with documentation.	Brief description of test and identification of RCTab functionalities tested by test files.	Unless otherwise noted, each regression test can be used to test control and data input/output, acceptance criteria, processing accuracy, ballot interpretation logic, and production of audit trails and statistical data. Processes for exporting and handling data, under control and data input, should also follow CVR handling procedures in a jurisdiction. Security processes should be tested according to the requirements in 06-NY System Security Specifications. Exception handling tests are directly identified below.
2013 Minneapolis Mayor	2013_minneapolis_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test RCTab's ability to properly count real-life RCV elections.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2013 Minneapolis Mayor Scale	2013_minneapolis_mayor_scale	Tests the ability of RCTab to process a contest with 1,000,000 records. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		CVR files.	
2013 Minneapolis Park	2013_minneapolis_park	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test bottoms-up multi-seat logic	2013_minneapolis_park_bottoms_up	Test the functionality of the bottoms-up winner election mode setting. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2017 Minneapolis Park	2017_minneapolis_park	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test Hare quota	2013_minneapolis_park_hare	Test the functionality of the Hare Quota Threshold Calculation Mode setting. Note that while RCTab passes this test the Hare Quota functionality needs to be updated as noted in 500-NY System Enhancements. Test the functionality of the "multi-winner allow multiple winners per round" functionality. Test the ability of RCTab to read and process ES&S	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		CVR files and CVR settings usable with ES&S CVR files.	
test sequential multi-seat logic	2013_minneapolis_park_sequential	Test the functionality of the multi-pass IRV winner election mode setting. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2015 Portland Mayor	2015_portland_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test ability to process a single-winner RCV contest.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2015 Portland Mayor Candidate Codes	2015_portland_mayor_codes	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files; ability to process a single-winner RCV contest; ability to process a CVR using Candidate Codes instead of Candidate Names	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2017 Minneapolis Mayor	2017_minneapolis_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files, as well as RCTab's ability to properly count real-life RCV elections.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2018 Maine Governor Democratic Primary	2018_maine_governor_primary	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files, as well as RCTab's ability	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		to properly count real-life RCV elections.	
Clear Ballot - Kansas Primary	clear_ballot_kansas_primary	Tests the ability of RCTab to read and process Clear Ballot CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Continue Until Two Candidates Remain	continue_tabulation_test	Tests the ability of RCTab to properly count a single-winner contest down to two final candidates	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Continue Until Two Candidates Remain with Batch Elimination	continue_until_two_with_batch_elimination_test	Tests the ability of RCTab to count a single-winner contest using both batch elimination and continue until two candidates remain settings simultaneously.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Alaska test data	dominion_alaska	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Kansas test data	dominion_kansas	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion - No Precinct Data	dominion_no_precinct_data	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Wyoming test data	dominion_wyoming	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		data.	
test excluding candidates in config file	excluded_test	Test the "Exclude" function in the Candidate settings of RCTab.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Hart - Cedar Park School Board	hart_cedar_park_school_board	Tests the ability of RCTab to read and process Hart CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Hart - Travis County Officers	hart_travis_county_officers	Tests the ability of RCTab to read and process Hart CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test invalid params in config file	invalid_params_test	Tests the ability of RCTab show errors if a configuration file is incomplete or improperly filled out.	Exception handling
test invalid source files	invalid_sources_test	Tests the ability of RCTab to show errors if CVR files are incompatible	Exception handling
test minimum vote threshold setting	minimum_threshold_test	Test the "minimum vote threshold" setting in winner election mode.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
testMinneapolisMultiSeatThreshold	minneapolis_multi_seat_threshold	Test the ability of RCTab to determine winning candidates according to the default Threshold Calculation method in multi-winner elections. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
missing precinct example	missing_precinct_example	Test the ability of RCTab to continue tabulating if "Precinct Column ID" is supplied but precinct	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		information is missing in part of a CVR file.	
test bottoms-up multi-seat with threshold logic	multi_seat_bottoms_up_with_threshold	Test the functionality of the bottoms-up using percentage threshold winner election mode setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat UWI test	multi_seat_uwi_test	Undeclared write-ins should not win elections. This test ensures that RCTab properly handles this exception by not awarding a win to an undeclared write-in candidate in a multi-winner contest.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
NIST XML CDF 2	nist_xml_cdf_2	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
precinct example	precinct_example	Test the ability of RCTab to detect precinct information based on CVR file data and the ability to produce precinct results using the Tabulate by Precinct functionality. Note that Tabulate by Precinct function produces precinct-level results of the RCV contest but does not identify precinct-level winners. This functionality needs updating to identify in-precinct winners.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

sample interactive tiebreak	sample_interactive_tiebreak	Test the functionality of the interactive tiebreak function available in "Stop counting and ask," or "Previous round counts (then stop counting and ask)" tiebreaking modes. This is not a regression test as it requires user input. It is included in the "sample_input" folder of RCTab installation folder as an additional test of RCTab.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test skipping to next candidate after overvote	skip_to_next_test	Test the functionality of the "Always skip to next rank" overvote setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
skipped first choice	test_set_0_skipped_first_choice	Test the ability of RCTab to properly process CVR data with a skipped first ranking. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
exhaust at overvote rule	test_set_1_exhaust_at_overvote	Test the functionality of the "Exhaust immediately" overvote setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
overvote skips to next rank	test_set_2_overvote_skip_to_next	Test the functionality of the "Always skip to next rank" overvote setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

skipped choice exhausts option	test_set_3_skipped_choice_exhaust	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots exhaust after a single skipped ranking. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
skipped choice next option	test_set_4_skipped_choice_next	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots don't exhaust after skipped rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
two skipped ranks exhausts option	test_set_5_two_skipped_choice_exhaust	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots exhaust after multiple skipped rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
duplicate rank exhausts	test_set_6_duplicate_exhaust	Test the functionality of the "Exhaust on multiple ranks for the same candidate" function, if function is turned on - ballots should exhaust when a duplicate ranking is encountered. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

duplicate rank skips to next option	test_set_7_duplicate_skip_to_next	Test the functionality of the "Exhaust on multiple ranks for the same candidate" function, if function is turned off - RCTab will ignore duplicate candidate rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-cdf tabulation	test_set_8_multi_cdf	Tests the ability of RCTab to read and process multiple CDF data files in XML format. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test allow only one winner per round logic	test_set_allow_only_one_winner_per_round	Test the functionality of the multi-winner allow only one winner per round logic.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat fractional number threshold	test_set_multi_winner_fractional_threshold	Test the functionality of the HB Quota Threshold Calculation Mode setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat whole number threshold	test_set_multi_winner_whole_threshold	Test the functionality of the default Threshold Calculation Mode setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
overvote delimiter test	test_set_overvote_delimiter	Test the functionality of the "overvote delimiter" setting. Note that this functionality needs an update as described in 500-NY System Enhancements.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

treat blank as undeclared write-in	test_set_treat_blank_as_undeclared_write_in	Test the functionality of the "Treat Blank as Undeclared Write-In" setting for CVRs.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
tiebreak using generated permutation	tiebreak_generate_permutation_test	Test the functionality of the "Generate permutation" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
tiebreak using previousRoundCountsThen Random	tiebreak_previous_round_counts_then_random_test	Test the functionality of the "Previous Round Counts then Random" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test tiebreak seed	tiebreak_seed_test	Test the functionality of the "Random seed" setting required for any of the Random Tiebreak modes ("Random" "Previous Round Counts then Random" and "Generate Permutation")	Control and data input/output; Processing accuracy; Ballot interpretation logic;
tiebreak using permutation in config	tiebreak_use_permutation_in_config	Test the functionality of the "Use candidate order in config file" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_chief_of_police	unisyn_xml_cdf_city_chief_of_police	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_coroner	unisyn_xml_cdf_city_coroner	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_council_member	unisyn_xml_cdf_city_council_member	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;

unisyn_xml_cdf_city_may or	unisyn_xml_cdf_city_mayor	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_tax_collector	unisyn_xml_cdf_city_tax_collector	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_county_coroner	unisyn_xml_cdf_county_coroner	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_county_sheriff	unisyn_xml_cdf_county_sheriff	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
undeclared write-in (UWI) cannot win test	uwi_cannot_win_test	Undeclared write-ins (UWIs) cannot win elections in RCTab tabulation. This test ensures that RCTab properly handles this exception by not awarding a win to an undeclared write-in candidate in a single-winner contest.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

Appendix

The Appendix contains the following files:

1. TabulatorTests_example_console_output
2. Google JAVA Style Guide

Appendix 7

RCTab Logic & Accuracy (L&A) Test Procedures

RCTab v.1.2.0 - 11 - L&A Testing v.1.2.0

The Logic & Accuracy Test is designed to verify that RCTab is correctly configured and operating properly. Logic & accuracy tests should always be conducted on RCTab prior to its use in an election as part of the user jurisdiction's full logic & accuracy tests for each election. Post-election logic & accuracy testing should also be performed where possible and in keeping with the established policies and procedures of the user jurisdiction.

MATERIALS REQUIRED. In addition to the following, you must know the correct version of your operating system and RCTab. The vendor can provide assistance in determining the correct version. Users can also find that information in the ***RCTab v.1.2.0 200 Installation Instructions for RCTab - Windows OS v.1.1.0*** documentation. The following components of a voting system are required to complete the acceptance testing procedures:

1. One Computer with RCTab installed. Installation instructions can be found in the ***RCTab v.1.2.0 200 Installation Instructions for RCTab - Windows OS v.1.1.0*** document.
2. General use directed connected printer (this includes necessary printer cable and power supply).
3. Recommended battery backup if user jurisdiction facility does not have generator capabilities in the event of a power failure.
4. One flash drive containing a CVR from your voting system with ranked-choice voting results.
5. One additional storage device that can be used for saving summary files.
6. Access to the Election Data Form ([see appendix](#)) or comparable user jurisdiction prepared form. This form may be completed digitally and may be required to be printed and stored as part of the election record.

The RCTab computer system, consisting of items 1 through 3 above, should have been set up and tested before beginning this test. IT staff may do this part of the setup if properly supervised by election staff who have been trained in accordance with procedures laid out in the personnel documentation.

VERIFY CORRECT OPERATING SYSTEM AND RCTab SOFTWARE.

Turn on the computer with RCTab installed. As the computer boots up, verify that the correct versions of the operating system and the RCTab software are installed. Compute the hash codes for the RCTab voting system software and compare them with the hash value provided with the trusted build. Refer to the ***RCTab v.1.2.0 03 System Hardware Specification v.1.1.0***.

document and the **RCTab v.1.2.0 200 Installation Instructions for Universal RCV Tabulator - Windows OS v.1.1.0** document for procedures and more information.

VERIFY RCTab HARDWARE.

Verify that all hardware components of the RCTab computer are turned on and functioning properly. This includes checks of the printer(s), flash drives, USB port, etc.

LOGIC & ACCURACY PROCEDURES - the procedures should be done with all assigned personnel in no less than bipartisan teams of two who have received the recommended vendor-led training. All vendor-recommended and jurisdiction-required security procedures should be observed at all times during this process.

1. **Assemble all required materials.** The next step will cover exporting a CVR from EMS to a flash drive if this step has not already been completed as part of the voting system L&A.
2. **Export CVR File from EMS System.** This step assumes the user has completed logic & accuracy checks on all parts of their voting system, including EMS, as recommended by the system vendor. This step also requires the users to have a known outcome from the test deck prepared as part of logic and accuracy testing as required by the voting system vendor. Users should also confirm the known outcome from the test deck is, in fact, correct, and the voting system used to tabulate the ballots is also working properly. The text deck should be counted by hand BEFORE the actual L&A testing begins on both the user jurisdiction's voting system and RCTab. This includes a headcount of the round-by-round RCV outcomes as well.
 - a. Following L&A testing procedures conducted in EMS, export the L&A CVR file from the jurisdiction's EMS System and save the file to an empty USB flash drive. Please see the appropriate EMS procedures for the specific steps of this process.
 - b. Take the USB containing the exported CVR to the hardware where the RCTab software is installed.
3. **Transfer the CVR File to the PC.**
 - a. Create a folder on the hard drive of the RCTab PC. The vendor recommends a name associated with the specific election (sample: "LA - RCV Election 11-5-2019"). The vendor recommends the user specifically identify the folder as LA to avoid any confusion with outdated CVRs or other data. See point 14, page 7 for further information.
 - b. Insert USB into the hardware where the RCTab software is installed and open the file with Excel
 - c. Copy the CVR export file from the USB to the new folder.
 - d. Enter the file path of the CVR file into the Election Data Form.
 - e. Remove the USB from the USB port and attach a label with the election name.
 - f. Place the USB in a secure storage location.

- 4. Enter Data into the Election Data Form.** Entering this information will make interacting with RCTab faster and lessen the chances that information will be entered incorrectly. All entries should be carefully reviewed by at least two bipartisan team members assigned to work with the RCTab software. Using your Election Data Form, record the following information:

- a. Contest Name: Name of the election – sample: LA-General Election, Primary Election, Municipal Election
- b. Contest Date: Date of the actual election – sample: 11-05-2019
- c. Contest Jurisdiction: Actual voting jurisdiction – sample: LA-Johnson County, City of Peoria, Town of Salem
- d. Contest Office: RCV office being tabulated – sample: LA-City Council, County Commissioner
- e. Decide if you are going to print Precinct Reports and enter Y or N on the Election Data Form.
- f. Indicate on the Election Data Form if you are generating a CDF JSON file for export totals (Y or N). Generally, this setting will be left unchecked as it is not required in most tabulations.
- g. Record the file path of the CVR as noted in step 3f, page 2, if you have not already done so.
- h. Enter the Rules into the Election Data Form The settings entered here must be in full compliance with the governing guidelines for ranked-choice voting elections in the jurisdiction. (check with City, County or State for guidance)

5. Open CVR File and Review

- a. Locate the CVR file in the election folder you created in step 3a and right-click, scroll down, and hover over “Open with.” Select Excel to view the file.
- b. Locate the Column where the First Vote is and enter the Column in the Election Data Form.
- c. Locate the Row where the First Vote is and enter the Row in the Election Data Form.
- d. Locate the ID Column (leave blank if not used) and enter the Column in the Election Data Form.
- e. Locate the Precinct Column and enter the Column in the Election Data Form
- f. Exit the CVR file.

Before moving to step 6, your Election Data Form should be completed with the exception of the Configuration File Name and required signatures. Confirm and enter any missing information before proceeding.

6. Creating the Configuration File

- a. Start RCTab - RCTab will start ready to create a new config file.
- b. Contest Info Tab

- i. On the tab labeled “Contest Info,” enter the Contest Name, Contest Date, Contest Jurisdiction, and Contest Office. For Rules Description, enter a name that is easy to remember as this can be used as the name for your final configuration file.
 - ii. Only the Contest Name is required for this tab; however, we recommend all fields be completed to assist in identifying output files in the future.
- c. CVR Files Tab
 - i. Select the voting system provider from the drop-down menu.
 - ii. Using the “Select” button, locate the CVR file in the election folder and load the CVR file path.
 - iii. Enter the remaining required information from the Election Data Form into the remaining fields and click Add.
 - iv. The added information should populate in the listing just below the entry fields.
- d. Candidates Tab
 - i. Click on the “Candidates” tab. Enter the first candidate name and press Add. Enter all candidates.
 - ii. If you are using Code, you can return to each name and enter Code, and press enter on the keyboard.
 - iii. You will not enter Exclude unless instructed to do so by the Election Administration Team.
- e. Winning Rules Tab
 - i. Use the completed Election Data Form to enter all applicable rule selections here.
 - ii. Depending on the required settings you enter, some options may remain grayed out and unselectable by the user. This is normal and should be expected as all fields may not require input.
 - iii. If there are any questions or concerns about the information needed to complete this tab, STOP and consult your controlling governing body for additional guidance about the jurisdiction’s particular ranked-choice voting requirements.
- f. Voter Error Rules
 - i. Use the completed Election Data Form to enter all applicable rules selections here.
 - ii. Depending on the required settings you enter, some options may remain grayed out and unselectable by the user. This is normal and should be expected as all fields may not require input.
 - iii. If there are any questions or concerns about the information needed to complete this tab, STOP and consult your controlling governing body for additional guidance about the jurisdiction’s particular ranked-choice voting requirements.

g. Output Tab

- i. In the output directory field, enter the file path to your election folder set up in 3a. This will place all tabulation files in this folder once the tabulation is complete.
- ii. If you are tabulating by precinct, check the box labeled “Tabulate by Precinct.”
- iii. If you are generating a CDF JSON, check the box labeled Generate CDF JSON. Generally, this setting will be left unchecked as it is not required in most tabulations.

7. Validating the Configuration File

- a. Click on “Tabulation” at the top of the software window.
- b. Click on “Validate”
- c. Refer to the log box at the bottom of the application. If the message “Contest config validation successful.” appears, your contest configuration has been successfully completed.
- d. If any error messages appear in the log box, refer to **RCTab v.1.2.0 430 Universal RCV Tabulator Operator Log Messages, v.1.1.0** and messages in the log box for how to resolve errors. If the error persists, restart the RCTab software.

8. Saving the Configuration File

- a. Click on “File” at the top of the software window.
- b. Click on “Save...”
- c. Select a location to save the configuration file. The vendor suggests users save the configuration file to the same location set in the Output Directory setting.
- d. Refer to the log box at the bottom of the application. If the message “Successfully saved file: Filepath” your configuration .json file has been successfully saved.
- e. If any error messages appear in the log box, refer to **RCTab v.1.2.0 430 Universal RCV Tabulator Operator Log Messages v.1.1.0** documentation and messages in the log box for how to resolve errors. If the error persists, restart RCTab software.

9. Once a configuration is saved, the user is ready to run a tabulation.

- a. Click on “Tabulation” at the top of the software window.
- b. Click on “Tabulate”
- c. Tabulation will begin.
- d. If all the above steps were successfully completed, tabulation will run until complete.
- e. The Tabulator log box will update with messages as Tabulation proceeds.
- f. Once complete, the Tabulator log box will display a message stating “Results written to: [filepath from Output Directory]”

10. Output files will be:

- a. .csv contest summary files

- i. Whole-contest summary files
 - b. .json contest summary files
 - i. Whole-contest summary files
 - c. .log audit files
 - i. .log audit files are exported in 50MB sections. If a .log file exceeds 50MB, an additional .log file is started by RCTab
- 11. Once output files have been generated, users should compare the results to the known test deck outcome prepared for the current election. If the results do not match, the settings entered into RCTab should be confirmed with officials that the configuration settings are in compliance with law and policy set by the user jurisdiction. Users should also confirm the known outcome from the test deck is, in fact, correct, and the voting system used to tabulate the ballots is also working properly. The text deck should be counted by hand BEFORE the actual L&A testing begins on both the user jurisdiction's voting system and RCTab. This includes a headcount of the round-by-round RCV outcomes as well.
- 12. Users can then navigate to "File" and click "Exit" if all contests are tabulated.
- 13. Users should produce and record hash codes for results files and configuration files from each successful use of RCTab.
 - a. Open the Start Menu
 - b. Type in Command Prompt. Press enter to launch the Command Prompt.
 - c. Type in certutil -hashfile [filename]
 - d. To insert the file name: locate the folder in Windows Explorer, including the files you want to produce hash files for.
 - e. Click on the file you want to produce a hash for, and while continuing to hold the mouse down, drag the file to the Command Prompt window and place after -hashfile.
 - i. Example: certutil -hashfile
C:\Users\user\Desktop\RCV-Results\2021-04-24_20-13-56_summary.csv
 - f. Type in SHA512 at the end of the line
 - i. Example: certutil -hashfile
C:\Users\user\Desktop\RCV-Results\2021-04-24_20-13-56_summary.csv
SHA512
 - g. Press enter. A hash file will appear in the Command Prompt.
 - h. Record this hash file value
- 14. If any errors arise in the use of RCTab, refer to the relevant documentation for the source of the error. RCTab errors should refer to **RCTab v. 1.2.0 430 Universal RCV Tabulator Operator Log Messages v.1.1.0**. Errors arising out of any hardware or software other than RCTab should refer to the RCTab Maintenance Procedures and any relevant user and maintenance manual.

15. We recommend that the configuration file be set up fully each time the user accesses RCTab. Starting a new configuration file each time will lessen the chances that outdated CVRs or other data is inadvertently introduced into the system.

Using Preloaded Sample CVRs to perform L&A

- As part of the download, RCTab includes three test CVR files along with configuration files and summary results. To test, the user access RCTab and should do the following:
 - Access the sample_input folder. This folder is part of the installation files created on the computer when RCTab was installed.
 - Choose the 2015_Portland_Mayor folder
 - Load the file 2015_portland_mayor_config.json
 - Load the file 2015_portland_mayor_cvr.xlsx
 - Validate the configuration file and run the tabulation
 - Compare the results to the summary file
2015_portland_mayor_expected_summary.json contained in the appendix of this document. If results do not match the user should confirm the correct configuration and CVR files were correctly selected and repeat the test. Any irreconcilable issues should be reported to the RCVRC for further examination.

Document Revision History

Date	Version	Description	Author
01/14/2022	1.2.0	Updated Names and edits for clarity	Ryan Kirby
05/14/2021	1.1.0	<ul style="list-style-type: none"> ● Updated to add information regarding creation of L&A test decks in steps 2 & 11 ● Added procedures for using sample CVRs to perform L&A 	Rosemary F. Blizzard
04/27/2021	1.0.0	L&A Testing Procedures	Rosemary F. Blizzard

2015_portland_mayor_expected_summary.pdf

```
{
  "config" : {
    "contest" : "Portland 2015 Mayoral Race", "date" :
"2015-11-03",
    "jurisdiction" : "Portland, ME", "office" :
"Mayor",
    "threshold" : "48"
  },
  "results" : [ {
    "round" : 1,
    "tally" : {
      "Bragdon, Charles E." : "11", "Brennan, Michael F."
: "6", "Bryant, Peter G." : "9",
      "Carmona, Ralph C." : "12",
      "Dodge, Richard A." : "10",
      "Duson, Jill C." : "5",
      "Eder, John M." : "3",
      "Haadoow, Hamza A." : "8",
      "Lapchick, Jodie L." : "7",
      "Marshall, David A." : "2",
      "Mavodones, Nicholas M. Jr." : "13", "Miller, Markos S."
: "3",
      "Rathband, Jed" : "2",
      "Strimling, Ethan K." : "1", "Undeclared Write-ins"
: "0", "Vail, Christopher L." : "6" },
    "tallyResults" : [ {
      "eliminated" : "Undeclared Write-ins", "transfers" : { }
    }, {
      "eliminated" : "Strimling, Ethan K.", "transfers" : {
      "Vail, Christopher L." : "1" }
    } ]
  }, {
    "round" : 2,
    "tally" : {
      "Bragdon, Charles E." : "11", "Brennan, Michael F."
: "6", "Bryant, Peter G." : "9",
      "Carmona, Ralph C." : "12",
      "Dodge, Richard A." : "10",
      "Duson, Jill C." : "5",
      "Eder, John M." : "3",
      "Haadoow, Hamza A." : "8",
```



```
"Lapchick, Jodie L." : "7",
"Marshall, David A." : "2",
"Mavodones, Nicholas M. Jr." : "13",  "Miller, Markos S."
: "3",
"Rathband, Jed" : "2",
"Vail, Christopher L." : "7"  },
"tallyResults" : [ {
"eliminated" : "Rathband, Jed",  "transfers" : {
"Mavodones, Nicholas M. Jr." : "2"  }
} ]
}, {
"round" : 3,
"tally" : {
"Bragdon, Charles E." : "11",  "Brennan, Michael F."
: "6",  "Bryant, Peter G." : "9",
"Carmona, Ralph C." : "12",  "Dodge, Richard A." :
"10",  "Duson, Jill C." : "5",
"Eder, John M." : "3",
"Haadoow, Hamza A." : "8",  "Lapchick, Jodie L." : "7",
"Marshall, David A." : "2",  "Mavodones, Nicholas M. Jr." :
"15",  "Miller, Markos S." : "3",  "Vail, Christopher L." :
"7"  },
"tallyResults" : [ {
"eliminated" : "Marshall, David A.",  "transfers" : {
"Miller, Markos S." : "2"  }
} ]
}, {
"round" : 4,
"tally" : {
"Bragdon, Charles E." : "11",  "Brennan, Michael F."
: "6",  "Bryant, Peter G." : "9",
"Carmona, Ralph C." : "12",  "Dodge, Richard A." :
"10",  "Duson, Jill C." : "5",
"Eder, John M." : "3",
"Haadoow, Hamza A." : "8",  "Lapchick, Jodie L." : "7",
"Mavodones, Nicholas M. Jr." : "15",  "Miller, Markos S." :
"5",  "Vail, Christopher L." : "7"  },
"tallyResults" : [ {
"eliminated" : "Eder, John M.",  "transfers" : {
"Duson, Jill C." : "3"
}
} ]
}, {
```

```

"round" : 5,
"tally" : {
  "Bragdon, Charles E." : "11",  "Brennan, Michael F."
: "6",  "Bryant, Peter G." : "9",
  "Carmona, Ralph C." : "12",
  "Dodge, Richard A." : "10",  "Duson, Jill C." :
"8",
  "Haadoow, Hamza A." : "8",
  "Lapchick, Jodie L." : "7",  "Mavodones, Nicholas M. Jr."
: "15",  "Miller, Markos S." : "5",
  "Vail, Christopher L." : "7"  },
"tallyResults" : [ {
  "eliminated" : "Miller, Markos S.",  "transfers" : {
  "Mavodones, Nicholas M. Jr." : "5"  }
} ]
}, {
"round" : 6,
"tally" : {
  "Bragdon, Charles E." : "11",  "Brennan, Michael F."
: "6",  "Bryant, Peter G." : "9",
  "Carmona, Ralph C." : "12",  "Dodge, Richard A." :
"10",  "Duson, Jill C." : "8",
  "Haadoow, Hamza A." : "8",
  "Lapchick, Jodie L." : "7",  "Mavodones, Nicholas M. Jr."
: "20",  "Vail, Christopher L." : "7"  },
"tallyResults" : [ {
  "eliminated" : "Brennan, Michael F.",  "transfers" : {
  "Bragdon, Charles E." : "3",  "Bryant, Peter G." :
"2",  "Carmona, Ralph C." : "1"  }
} ]
}, {
"round" : 7,
"tally" : {
  "Bragdon, Charles E." : "14",  "Bryant, Peter G." :
"11",
  "Carmona, Ralph C." : "13",  "Dodge, Richard A." :
"10",  "Duson, Jill C." : "8",
  "Haadoow, Hamza A." : "8",
  "Lapchick, Jodie L." : "7",  "Mavodones, Nicholas M. Jr."
: "20",  "Vail, Christopher L." : "7"  },
"tallyResults" : [ {
  "eliminated" : "Vail, Christopher L.",  "transfers" : {
  "Mavodones, Nicholas M. Jr." : "6",  "exhausted" : "1"

```

```

}
} ]
}, {
  "round" : 8,
  "tally" : {
    "Bragdon, Charles E." : "14", "Bryant, Peter G." :
    "11", "Carmona, Ralph C." : "13", "Dodge, Richard
    A." : "10", "Duson, Jill C." : "8",
    "Haadoow, Hamza A." : "8", "Lapchick, Jodie L." : "7",
    "Mavodones, Nicholas M. Jr." : "26" },
    "tallyResults" : [ {
      "eliminated" : "Lapchick, Jodie L.", "transfers" : {
        "Haadoow, Hamza A." : "1", "Mavodones, Nicholas M. Jr." :
        "6" }
    } ]
  }, {
    "round" : 9,
    "tally" : {
      "Bragdon, Charles E." : "14", "Bryant, Peter G." :
      "11", "Carmona, Ralph C." : "13", "Dodge, Richard
      A." : "10", "Duson, Jill C." : "8",
      "Haadoow, Hamza A." : "9", "Mavodones, Nicholas M. Jr."
      : "32" },
      "tallyResults" : [ {
        "eliminated" : "Duson, Jill C.", "transfers" : {
          "Bryant, Peter G." : "1", "Dodge, Richard A." : "5",
          "Mavodones, Nicholas M. Jr." : "2" }
        } ]
    }, {
      "round" : 10,
      "tally" : {
        "Bragdon, Charles E." : "14", "Bryant, Peter G." :
        "12", "Carmona, Ralph C." : "13", "Dodge, Richard A." :
        "15", "Haadoow, Hamza A." : "9", "Mavodones, Nicholas M.
        Jr." : "34" },
        "tallyResults" : [ {
          "eliminated" : "Haadoow, Hamza A.", "transfers" : {
            "Carmona, Ralph C." : "2", "Mavodones, Nicholas M. Jr." :
            "7" }
        } ]
      }, {
        "round" : 11,
        "tally" : {

```

```

"Bragdon, Charles E." : "14",
"Bryant, Peter G." : "12",
"Carmona, Ralph C." : "15",
"Dodge, Richard A." : "15",
"Mavodones, Nicholas M. Jr." : "41" },
"tallyResults" : [ {
"eliminated" : "Bryant, Peter G.", "transfers" : {
"Bragdon, Charles E." : "9", "Carmona, Ralph C." :
"2",
"Mavodones, Nicholas M. Jr." : "1" }
} ]
}, {
"round" : 12,
"tally" : {
"Bragdon, Charles E." : "23",
"Carmona, Ralph C." : "17",
"Dodge, Richard A." : "15",
"Mavodones, Nicholas M. Jr." : "42" },
"tallyResults" : [ {
"eliminated" : "Dodge, Richard A.", "transfers" : {
"Bragdon, Charles E." : "1", "Carmona, Ralph C." :
"11",
"Mavodones, Nicholas M. Jr." : "3" }
} ]
}, {
"round" : 13,
"tally" : {
"Bragdon, Charles E." : "24",
"Carmona, Ralph C." : "28",
"Mavodones, Nicholas M. Jr." : "45" },
"tallyResults" : [ {
"eliminated" : "Bragdon, Charles E.", "transfers" : {
"Carmona, Ralph C." : "12",
"Mavodones, Nicholas M. Jr." : "9", "exhausted" : "3"
}
} ]
}, {
"round" : 14,
"tally" : {
"Carmona, Ralph C." : "40",
"Mavodones, Nicholas M. Jr." : "54" },
"tallyResults" : [ {
"elected" : "Mavodones, Nicholas M. Jr.", "transfers" : { }

```

}]
}]
}

Appendix 8

Quality Assurance Plan

RCTAB v.1.2.0 13 - Quality Assurance Plan v.1.2.0

Scope:

This document outlines the general Quality Assurance processes and procedures of the RCTAB counting software.

The RCTAB software is designed for use as a round-by-round counting software and installed on COTS equipment after centralization of the cast vote record data. Centralization of data occurs based on the policies, procedures, and laws of the jurisdiction using the counting software. This software is designed specifically for use in ranked-choice voting elections and for the purpose of New York City will be used for counting single-winner contests.

The software is simple to install and utilize with proper training of election officials. From the earliest concept of this product, RCTAB was designed to provide confidence and accuracy in the round-by-round count of any jurisdiction. The software is designed for election official input based on jurisdiction RCV rules. The software can use CVR data from Dominion, ES&S, Hart, and Unisyn voting systems.

Items Covered in the Scope:

- Requirements, design process, and definition of the RCTAB software.
- Determination of the specifications that a COTS device must meet in order to optimize RCTAB installation and operation.
- The process recommendations for centralization of CVR data prior to round-by-round counting.
- Validation and verification of the performance of the RCTAB.
- Validation and verification of the process for installation of RCTAB on COTS hardware along with the necessary steps to secure the system as would be required in a jurisdiction.

Requirements, Design Process, and Definition of RCTAB Software:

Requirements:

The RCTAB is tabulation software that is designed to use voting system data (such as the Cast Vote Record (CVR) from ES&S voting equipment) to run a round-by-round tabulation of a ranked-choice voting contest. The software must be installed on a computer configured according to the specifications listed in our **RCTAB 1.2.0 - 3 - System Hardware Specification v.1.2.0**.

The computer must be a standalone computer that is not connected to an internet connection or a network of any kind. Wireless connection must be disabled if available on the computer. All industry standard security configurations must be set up and any security policies must be followed, included but not limited to: computer hardening and installation of a recommended antivirus software. For additional information, see documentation listed below.

Description of Parts and Materials Necessary for RCTAB Use: (V.2: 2.12.2--Description)

- Trusted build of RCTAB to be acquired from SLI or the NY State Board of Elections.
- COTS computer that is not connected to the internet and is configured according to all specifications laid out in the following documents: **RCTAB 1.2.0 NY-3 System Hardware Specification 1.1.0**, **RCTAB 1.2.0 NY-16 System Hardening Procedures 1.1.0**, **RCTAB 1.2.0 12 Configuration Management Plan**, and **RCTAB 1.2.0 NY-7 System Security Specification Requirements 1.1.0**.
- Battery backup is also recommended if the facility does not have a generator. This is not required to run the software.

Ensuring Proper Functionality of the Parts and Materials:

- The parts were chosen in the following manner in accordance with VVSG Volume 2: 2:12.1 as it refers to Volume 1:8.5 and Volume 1:8.6.
 - RCTAB software--the voting tabulation product. (See below for all testing information regarding the product.)
 - COTS hardware was chosen as an industry standard for a Windows machine. All set up should refer to the documentation listed above. RCVRC does not design, manufacture, or resale any hardware.
- Test data storage process has been updated to meet the specifications outlined in the VVSG Volume 1:8.5c

Design Process:

The RCTAB software is developed with the following tools, policies, and practices to ensure robust software quality and reliability. RCTAB testing relates only to the function of the software and performs no parts and materials testing as we produce only software and do not design or manufacture hardware. Testing follows standards laid out in the VVSG Volume 2:2.12.1 with regard to V1:8.5.

The **RCTAB 1.2.0 04 System Functionality Description v.1.2.0** incorporates general functional requirements for the tabulation software system. Requirements for accuracy were taken into account when designing and performing all testing including stress tests. The VVSG 2.0 requirements with regard to accuracy were also utilized when designing and performing stress tests. V2:2.12.1 also refers to V1:8.5, V1:8.6 and V1:8.7 and were followed throughout the process. Regression testing is completed for each design change or addition, whether minor or major. For additional information about testing, refer to **RCTAB 1.2.0, 17 System Test and Verification Specification**.

Design, Testing, and QA Process Responsibilities:

The design process and QA assessments are performed by a joint team that includes the developers from Bright Spots and software team from EARC/RCVRC. Project managers may be used on a contract basis to manage QA responsibilities such as testing.

Documentation of Quality Conformance Procedures:

The RCTAB development team uses a ticketing system to identify bugs as well as design issues. All procedures below reflect this process and are managed through the Resource Center and Bright Spots using GitHub. Refer to the **RCTAB 1.2.0 12 Configuration Management Plan** to understand the process more fully.

QA Processes and Procedures:

1) Version Control Software:

We use Git version control software (git-scm.com) in conjunction with Github (github.com/BrightSpots/rcv) to coordinate the efforts of our developers and maintain a complete record of ALL software code changes to the RCTAB and the reasoning behind them.

2) Software Revision Control Branching Policy:

To isolate code in development from production code (released), we use Git-Flow: a branching policy built on top of Git. The policy coordinates development, testing, review, and deployment of code throughout the development life-cycle. Details can be found here: (www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow)

3) Code reviews:

All code changes submitted for incorporation into the RCTAB software must undergo a manual code review from at least one developer other than the original drafter/developer. Other stakeholders and experts are involved with code reviews as needed. Code reviews offer an additional opportunity to identify potential defects, improve code structure, clarity, performance, and robustness. Code merges are blocked until at least one developer explicitly approves the code submission, at which point the code is merged, and regression tests will be run. For code review examples, see: github.com/BrightSpots/rcv/pulls.

4) Regression Tests:

We have developed a suite of 57 Tabulation Regression Tests. We continue to add more tests as new features and bug fixes are added. These tests are designed to verify that new code changes do not inadvertently alter any tabulation results.

Each test has a set of inputs (just like any tabulation): A config file and CVR file(s). Additionally, each test has an expected results file. The automated test will: load the config, run the tabulation, verify that the tabulation output matches the expected results. For actual test data, see: github.com/BrightSpots/rcv/tree/master/src/test/resources/network/brightspots/rcv/test_data.

The names of the tests are mostly self-explanatory. For example, `test_set_1_exhaust_at_overvote` tests whether the tabulator correctly interprets the

`overvoteRule = "exhaust immediately"` setting. Some of the tests aren't testing specific features but instead are testing a known data set, e.g., `2018_maine_governor_primary`. The complete list is included below.

All regression tests are run by developers and must pass before any new code can be merged into the tabulator. See Appendix A-1 for list of regression tests.

Quality Conformance Inspections and Documentation:

1) System Requirements Testing:

Minimum system requirements were determined by repeated tabulation of elections with 100,000 CVRs, 1,000,000 CVRs, and 6,000,000 CVRs on each target platform. These tests were timed to ensure they complete in a reasonable amount of time successfully.

2) Ad-hoc user testing:

When developing new features, we try to recruit as much user feedback and user testing as possible. These testers are typically from the RCVRC staff, vendor partners, election administrators, and RCV activists.

3) *Process for Handling of deficiencies:*

Any defects discovered through testing or reported by users are recorded and tracked using Github issue tracking tools. We confirm the existence of the defect, evaluate its severity, and mark it accordingly. Depending on user impact, development resources, and release timelines, we schedule developers to address the defects in order of priority and then do the actual work.

Typically, this involves:

1. reproducing the defect
2. making necessary code changes to fix the defect on an isolated git branch
3. requesting a code review and implementing any changes arising from the code review
4. verifying that regression tests pass
5. pushing the code change to the main branch and linking the issue in the commit message
6. closing the issue and linking to the commit in Github issue tracker

In this way, we are able to ensure that all known issues are tracked, and the most important ones are mitigated according to criticality. For more details, see:

<https://github.com/BrightSpots/rcv/issues>.

Testing Documentation:

A timeline of testing that has occurred to date can be located in the **RCTAB 1.2.0 7 System Test and Verification Specification**. We have updated general procedures moving forward to include a test documentation log with specific testing details including:

- Test Date
- Type and description of test performed
- Location test performed
- Individual/individuals who conducted the test
- Test Outcomes

Regression Test Overview:

We have developed a suite of 57 Tabulation Regression Tests and continue to add more tests as new features and bug fixes are added. These tests are designed to verify various aspects of Tabulator functionality behave as expected. They also verify that new code changes do not inadvertently alter Tabulator behavior. The entire test suite must be run, and all tests must pass, before any new code changes can be merged into the main Tabulator repository.

Design Overview:

Each test contains a set of normal tabulation inputs (config file and cvr files) and a known valid "expected" results summary file. The test runs a tabulation with new code, and compares the results of that tabulation to previous, expected, correct results. In essence, we isolate and analyze the effects any new code changes may have on the tabulation output. This is a classic regression test design.

Test Execution Details:

The test suite will run through all tests automatically as follows:

1. Tabulator is built from source code.
2. For each test:

- a. Run a tabulation using the test config file and cvr files.
- b. Compare tabulation output summary file to reference expected summary file.
- c. If the files match exactly (except for timestamps) the test passes.
- d. If the files do not match the test fails.
- e. Test execution and test results are written to a log file and console.

Testing Procedures:

When new code is ready for submission a developer will follow these procedures to ensure the code is safe for incorporation:

1. Run test suite: in a console, from the rcv root directory, enter: `./gradlew.bat test`
2. Observe the test output. If *any* test fails the source of the failure must be identified and either:
 - a. Fixed. Usually, a test failure is caused by a bug in logic or data which can be fixed.
 - b. Update the reference test asset. Sometimes bug fixes cause tests to fail because they are now operating correctly. In these cases, test output must be manually verified and peer reviewed (like any code change) before it can be updated.

Example test outputs are included in: `TabulatorTests_example_console_output.txt` and `Test_Results_TabulatorTests.html`

For actual test data, including configuration files and expected results, see: github.com/BrightSpots/rcv/tree/master/src/test/resources/network/brightspots/rcv/test_data. To download the data, go to this page: <https://github.com/BrightSpots/rcv/tree/v1.2.0>. This is the location on GitHub where version 1.2.0 of the RCTAB source code is hosted. Click on “Code” and select “Download ZIP.” Once the ZIP is downloaded, extract it. Navigate to the unzipped folder and navigate to `src/test/resources/network/brightspots/rcv/test_data`. All test data files will be included in this folder.

Users can also manually run each test. Steps required to run tests manually are as follows:

1. Open RCTAB
2. Click “File”
3. Click “Load”
4. Navigate to the configuration file for the test you wish to run
5. Select the configuration file and load it into the RCTAB
6. Confirm that the configuration file properly loaded into the RCTAB by checking that the relevant fields are filled in
7. Click “Tabulation”
8. Click “Tabulate”
9. Wait for the RCTAB to finish tabulating
10. Navigate to the output location for your results
11. Compare your results .json summary file to the .json summary file included with the test data. If these match exactly, the test passes.

Additional Testing Procedures:

In addition to regression testing of all changes, new features are tested by developers and

RCVRC staff to ensure that they work as expected. If any features did not work as expected, a ticket was filed on GitHub with the test data and an explanation of how the achieved results differed from the expected results.

The RCVRC and Bright Spots also conduct scale tests of the RCTab. Tests include a contest with 100 CVR files composed of 100,000 individual cast vote records each, a contest with 1,000 CVR files composed of 1,000,000 individual cast vote records each, a contest with 6,000 CVR files composed of 6,000,000 individual cast vote records each, and a set of 11 CVRs composed of 9,200,000 individual cast vote records each. Volume tests with these records were conducted throughout March and April 2021.

Additionally, see List of Tests at the end of this document starting at p. 9.

All documentation in this submission is named according to the formula:

[System version] [Document Number-NY] [Document Name] [Document version] System Version: RCTAB 1.2.0

Document version: 1.0.0 (all documents are new with this submission)

Example:

RCTAB 1.2.0 11 Quality Assurance Plan 1.0.0

Product Documentation:

RCTAB provides documentation in order to meet the VVSG standards outlined in Volume 2, Section 2, Description of the Technical Data Package. Refer to this reference list of documentation for more information:

• RCTAB 1.2.0, 01 System Overview
• RCTAB 1.2.0, 02 Software Design and Specifications
• RCTAB 1.2.0, 03 System Hardware Specification
• RCTAB 1.2.0, 04 System Functionality Description
• RCTAB 1.2.0, 07 System Security Specification Requirements
• RCTAB 1.2.0, 08 System Operations Procedures
• RCTAB 1.2.0, 09 System Maintenance Procedures
• RCTAB 1.2.0, 10 Personnel Deployment and Training
• RCTAB 1.2.0, 12 Configuration Management Plan
• RCTAB 1.2.0, 13 Quality Assurance Plan
• RCTAB 1.2.0, 15 System Change Notes

<ul style="list-style-type: none"> • RCTAB 1.2.0, 16 System Hardening Procedures
<ul style="list-style-type: none"> • RCTAB 1.2.0, 17 System Test and Verification Specification
<ul style="list-style-type: none"> • RCTAB 1.2.0, 18 User Guide

Appendix A:

RCTAB Regression Test Listing:

@DisplayName("NIST XML CDF 2")
 @DisplayName("unisyn_xml_cdf_city_tax_collector")
 @DisplayName("unisyn_xml_cdf_city_mayor")
 @DisplayName("unisyn_xml_cdf_city_council_member")
 @DisplayName("unisyn_xml_cdf_city_chief_of_police")
 @DisplayName("unisyn_xml_cdf_city_coroner")
 @DisplayName("unisyn_xml_cdf_county_sheriff")
 @DisplayName("unisyn_xml_cdf_county_coroner")
 @DisplayName("Clear Ballot - Kansas Primary")
 @DisplayName("Hart - Travis County Officers")
 @DisplayName("Hart - Cedar Park School Board")
 @DisplayName("Dominion test - Alaska test data")
 @DisplayName("Dominion test - Kansas test data")
 @DisplayName("Dominion test - Wyoming test data")
 @DisplayName("Dominion - No Precinct Data")
 @DisplayName("test invalid params in config file")
 @DisplayName("test invalid source files")
 @DisplayName("2015 Portland Mayor")
 @DisplayName("2015 Portland Mayor Candidate Codes")
 @DisplayName("2013 Minneapolis Mayor Scale")
 @DisplayName("Continue Until Two Candidates Remain")
 @DisplayName("Continue Until Two Candidates Remain with Batch Elimination")
 @DisplayName("2017 Minneapolis Mayor")
 @DisplayName("2013 Minneapolis Mayor")
 @DisplayName("2013 Minneapolis Park")
 @DisplayName("2018 Maine Governor Democratic Primary")
 @DisplayName("testMinneapolisMultiSeatThreshold")
 @DisplayName("test for overvotes")
 @DisplayName("test excluding candidates in config file")
 @DisplayName("test minimum vote threshold setting")
 @DisplayName("test skipping to next candidate after overvote") @DisplayName("test Hare quota")
 @DisplayName("test sequential multi-seat logic")

@DisplayName("test bottoms-up multi-seat logic")
 @DisplayName("test bottoms-up multi-seat with threshold logic") @DisplayName("test allow only one winner per round logic")
 @DisplayName("precinct example")
 @DisplayName("missing precinct example")
 @DisplayName("test tiebreak seed")
 @DisplayName("skipped first choice")
 @DisplayName("exhaust at overvote rule")
 @DisplayName("overvote skips to next rank")
 @DisplayName("skipped choice exhausts option")
 @DisplayName("skipped choice next option")
 @DisplayName("two skipped ranks exhausts option")
 @DisplayName("duplicate rank exhausts")
 @DisplayName("duplicate rank skips to next option")
 @DisplayName("multi-cdf tabulation")
 @DisplayName("multi-seat whole number threshold")
 @DisplayName("multi-seat fractional number threshold")
 @DisplayName("tiebreak using permutation in config")
 @DisplayName("tiebreak using generated permutation")
 @DisplayName("tiebreak using previousRoundCountsThenRandom")
 @DisplayName("treat blank as undeclared write-in")
 @DisplayName("undeclared write-in (UWI) cannot win test")
 @DisplayName("multi-seat UWI test")
 @DisplayName("overvote delimiter test")

Date	Version	Description	Author
05/9/2021	1.1.0	Updates to scope of work, reference documents, and parts and Materials	Kelly Sechrist
05/02/2021	1.0.0	Quality Assurance Plan	Chris Hughes

List of Tests

Name of Test	Name of Test Folder	Description of test	Purpose of Test
RCVRC & Bright Spots name for regression test	<u>Test folder name in:</u> Refer to test_data.zip that includes all test data and was submitted with documentation.	Brief description of test and identification of RCTab functionalities tested by test files.	Unless otherwise noted, each regression test can be used to test control and data input/output, acceptance criteria, processing accuracy, ballot interpretation logic, and production of audit trails and statistical data. Processes for exporting and handling data, under control and data input, should also follow CVR handling procedures in a jurisdiction. Security processes should be tested according to the requirements in 06-NY System Security Specifications. Exception handling tests are directly identified below.
2013 Minneapolis Mayor	2013_minneapolis_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test RCTab's ability to properly count real-life RCV elections.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2013 Minneapolis Mayor Scale	2013_minneapolis_mayor_scale	Tests the ability of RCTab to process a contest with 1,000,000 records. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

2013 Minneapolis Park	2013_minneapolis_park	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test bottoms-up multi-seat logic	2013_minneapolis_park_bottoms_up	Test the functionality of the bottoms-up winner election mode setting. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2017 Minneapolis Park	2017_minneapolis_park	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test Hare quota	2013_minneapolis_park_hare	Test the functionality of the Hare Quota Threshold Calculation Mode setting. Note that while RCTab passes this test the Hare Quota functionality needs to be updated as noted in 500-NY System Enhancements. Test the functionality of the "multi-winner allow multiple winners per round" functionality. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

test sequential multi-seat logic	2013_minneapolis_park_sequential	Test the functionality of the multi-pass IRV winner election mode setting. Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2015 Portland Mayor	2015_portland_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files. Test ability to process a single-winner RCV contest.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2015 Portland Mayor Candidate Codes	2015_portland_mayor_codes	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files; ability to process a single-winner RCV contest; ability to process a CVR using Candidate Codes instead of Candidate Names	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2017 Minneapolis Mayor	2017_minneapolis_mayor	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files, as well as RCTab's ability to properly count real-life RCV elections.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
2018 Maine Governor Democratic Primary	2018_maine_governor_primary	Test the ability of RCTab to read and process ES&S CVR files and CVR settings usable with ES&S CVR files, as well as RCTab's ability to properly count real-life RCV elections.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Clear Ballot - Kansas Primary	clear_ballot_kansas_primary	Tests the ability of RCTab to read and process Clear Ballot CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;

Continue Until Two Candidates Remain	continue_tabulation_test	Tests the ability of RCTab to properly count a single-winner contest down to two final candidates	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Continue Until Two Candidates Remain with Batch Elimination	continue_until_two_with_batch_elimination_test	Tests the ability of RCTab to count a single-winner contest using both batch elimination and continue until two candidates remain settings simultaneously.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Alaska test data	dominion_alaska	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Kansas test data	dominion_kansas	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion - No Precinct Data	dominion_no_precinct_data	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Dominion test - Wyoming test data	dominion_wyoming	Tests the ability of RCTab to read and process Dominion CVR data and to run an RCV count on that data.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test excluding candidates in config file	excluded_test	Test the "Exclude" function in the Candidate settings of RCTab.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
Hart - Cedar Park School Board	hart_cedar_park_school_board	Tests the ability of RCTab to read and process Hart CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;

Hart - Travis County Officers	hart_travis_county_officers	Tests the ability of RCTab to read and process Hart CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test invalid params in config file	invalid_params_test	Tests the ability of RCTab show errors if a configuration file is incomplete or improperly filled out.	Exception handling
test invalid source files	invalid_sources_test	Tests the ability of RCTab to show errors if CVR files are incompatible	Exception handling
test minimum vote threshold setting	minimum_threshold_test	Test the "minimum vote threshold" setting in winner election mode.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
testMinneapolisMultiSeatThreshold	minneapolis_multi_seat_threshold	Test the ability of RCTab to determine winning candidates according to the default Threshold Calculation method in multi-winner elections. Test the functionality of the "multi-winner allow multiple winners per round" functionality.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
missing precinct example	missing_precinct_example	Test the ability of RCTab to continue tabulating if "Precinct Column ID" is supplied but precinct information is missing in part of a CVR file.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test bottoms-up multi-seat with threshold logic	multi_seat_bottoms_up_with_threshold	Test the functionality of the bottoms-up using percentage threshold winner election mode setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat UWI test	multi_seat_uwi_test	Undeclared write-ins should not win elections. This test ensures that RCTab properly handles this exception by not awarding a win to an undeclared write-in candidate in	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		a multi-winner contest.	
NIST XML CDF 2	nist_xml_cdf_2	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
precinct example	precinct_example	Test the ability of RCTab to detect precinct information based on CVR file data and the ability to produce precinct results using the Tabulate by Precinct functionality. Note that Tabulate by Precinct function produces precinct-level results of the RCV contest but does not identify precinct-level winners. This functionality needs updating to identify in-precinct winners.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
sample interactive tiebreak	sample_interactive_tiebreak	Test the functionality of the interactive tiebreak function available in "Stop counting and ask" or "Previous round counts (then stop counting and ask)" tiebreaking modes. This is not a regression test as it requires user input. It is included in the "sample_input" folder of RCTab installation folder as an additional test of RCTab.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test skipping to next candidate after overvoted	skip_to_next_test	Test the functionality of the "Always skip to next rank" overvoted setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
skipped first choice	test_set_o_skipped_first_choice	Test the ability of RCTab to properly process CVR data with a skipped first ranking. Test the ability of RCTab to export CVR data	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		in the CDF.	
exhaust at overvote rule	test_set_1_exhaust_at_overvote	Test the functionality of the "Exhaust immediately" overvote setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
overvote skips to next rank	test_set_2_overvote_skip_to_next	Test the functionality of the "Always skip to next rank" overvote setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
skipped choice exhausts option	test_set_3_skipped_choice_exhaust	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots exhaust after a single skipped ranking. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
skipped choice next option	test_set_4_skipped_choice_next	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots don't exhaust after skipped rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
two skipped ranks exhausts option	test_set_5_two_skipped_choice_exhaust	Test the functionality of the "how many consecutive skipped ranks are allowed" setting when ballots exhaust after multiple skipped rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
duplicate rank exhausts	test_set_6_duplicate_exhaust	Test the functionality of the "Exhaust on multiple ranks for the same candidate" function, if function is turned on - ballots should exhaust when a duplicate ranking is encountered. Test the	Control and data input/output; Processing accuracy; Ballot interpretation logic;

		ability of RCTab to export CVR data in the CDF.	
duplicate rank skips to next option	test_set_7_duplicate_skip_to_next	Test the functionality of the "Exhaust on multiple ranks for the same candidate" function, if function is turned off - RCTab will ignore duplicate candidate rankings. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-cdf tabulation	test_set_8_multi_cdf	Tests the ability of RCTab to read and process multiple CDF data files in XML format. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test allow only one winner per round logic	test_set_allow_only_one_winner_per_round	Test the functionality of the multi-winner allow only one winner per round logic.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat fractional number threshold	test_set_multi_winner_fractional_threshold	Test the functionality of the HB Quota Threshold Calculation Mode setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
multi-seat whole number threshold	test_set_multi_winner_whole_threshold	Test the functionality of the default Threshold Calculation Mode setting. Test the ability of RCTab to export CVR data in the CDF.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
overvote delimiter test	test_set_overvote_delimiter	Test the functionality of the "overvote delimiter" setting. Note that this functionality needs an update as described in 500-NY System Enhancements.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
treat blank as undeclared write-in	test_set_treat_blank_as_undeclared_write_in	Test the functionality of the "Treat Blank as Undeclared Write-In"	Control and data input/output; Processing accuracy;

		setting for CVRs.	Ballot interpretation logic;
tiebreak using generated permutation	tiebreak_generate_permutation_test	Test the functionality of the "Generate permutation" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
tiebreak using previousRoundCountsThen Random	tiebreak_previous_round_counts_then_random_test	Test the functionality of the "Previous Round Counts then Random" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
test tiebreak seed	tiebreak_seed_test	Test the functionality of the "Random seed" setting required for any of the Random Tiebreak modes ("Random" "Previous Round Counts then Random" and "Generate Permutation")	Control and data input/output; Processing accuracy; Ballot interpretation logic;
tiebreak using permutation in config	tiebreak_use_permutation_in_config	Test the functionality of the "Use candidate order in config file" tiebreak setting.	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_chief_of_police	unisyn_xml_cdf_city_chief_of_police	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_coroner	unisyn_xml_cdf_city_coroner	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_council_member	unisyn_xml_cdf_city_council_member	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_mayor or	unisyn_xml_cdf_city_mayor	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_city_tax_collector	unisyn_xml_cdf_city_tax_collector	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;

unisyn_xml_cdf_county_coroner	unisyn_xml_cdf_county_coroner	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
unisyn_xml_cdf_county_sheriff	unisyn_xml_cdf_county_sheriff	Tests the ability of RCTab to read and process CDF data in XML format	Control and data input/output; Processing accuracy; Ballot interpretation logic;
undeclared write-in (UWI) cannot win test	uwi_cannot_win_test	Undeclared write-ins (UWIs) cannot win elections in RCTab tabulation. This test ensures that RCTab properly handles this exception by not awarding a win to an undeclared write-in candidate in a single-winner contest.	Control and data input/output; Processing accuracy; Ballot interpretation logic;

