RCTab Documentation

v1.3.2

The Ranked Choice Voting Resource Center

Table of contents

1. RCTab v1.3.2 Docs 4
2. TDP 5
2.1 Documentation Abstracts 5
2.2 Section 01 - System Overview 9
2.3 Section 02 - Software Design and Specifications 10
2.4 Section 03 - System Hardware Specification 27
2.5 Section 04 - System Functionality Description 29
2.6 Section 05 - Acceptance Test Procedures 31
2.7 Section 06 - System Design Specifications 32
2.8 Section 07 - System Security Specification Requirements 33
2.9 Section 07I - Design and Interface Specification 44
2.10 Section 07J - Security Architecture 46
2.11 Section 07K - Development Environment Specification 47
2.12 Section 07L - Security Threat Analysis 48
2.13 Section 07M - Security Testing and Vulnerability Analysis 51
2.14 Section 09 - System Maintenance Manual 52
2.15 Section 10 - Personnel Deployment and Training 57
2.16 Section 11 - L&A Testing 59
2.17 Section 12 - Configuration Management Plan 67
2.18 Section 13 - Quality Assurance Plan 74
2.19 Section 14 - Tabulator Trusted Build Instructions 77
2.20 Section 15 - System Change Notes 87
2.21 Section 16 - System Hardening Procedures - Windows OS 95
2.22 Section 17 - System Test and Verification Specification 112
2.23 Section 18 - User Guide 142
2.24 Section 19 - Tabulation Options for RCV Tabulation 170
2.25 Section 20 - Process Ranked Choice Voting Contest 185
2.26 Section 21 - Ballot Limitations & Maximum Testing Range 192
2.27 Section 22 - Installation Instructions for Windows OS 193
2.28 Section 23 - Trusted Build & Output Hash Verification - Windows OS 198
2.29 Section 24 - Tabulator Command Line Instructions 199
2.30 Section 25 - Configuration File Parameters 205
2.31 Section 26 - RCTab CVR Files 216
2.32 Section 27 - RCTab Config Files 226
2.33 Section 28 - Post-Election Audit & Clearing RCTab from System 227

2.34 Section 29 - RCTab Operator Log Messages 233
2.35 Section 30 - RCTab System Tab Hints 242
2.36 Section 31 - Coding and Header Comment Standards for RCTab 247
2.37 Section 32 - Secure USB Process 248
2.38 Appendix I - Expected Outcome RCV Test Sets Multi-Winner 252
2.39 Appendix II - Expected Outcome RCV Test Sets Single-Winner 253
2.40 Appendix III - Ranked Choice Voting Laws 254
2.41 Appendix IV - 22-Month Archiving Procedure 255
3. User Guide 256
3.1 User Guide 256

1. RCTab v1.3.2 Docs

RCTab is an open source tabulator for running ranked choice voting elections.

- **User Guide** Start here if you are a new user. It will help you get RCTab configured and running tabulations.
- **TDP** The Technical Documentation Package or TDP contains detailed documentation to satisfy state and federal certification requirements.

2. TDP

2.1 Documentation Abstracts

Section 01 - System Overview

The System Overview provides a brief introduction to the RCTab software and its intended use cases.

Section 02 - Software Design and Specifications

The Software Design and Specifications document describes RCTab coding standards, programming language, in-house software, third-party software, and software logic. It is designed to respond to CVSS 9.5.

Section 03 - System Hardware Specification

The System Hardware Specification document describes potential hardware configurations on which the RCTab system can run.

Section 04 - System Functionality Description

The System Functionality Description describes suggested procedures for recovering from potential hardware or software failures when using an RCTab workstation.

Section 05 - Acceptance Test Procedures

Acceptance Test Procedures verify that RCTab is correctly configured and operating properly on an RCTab workstation. Acceptance tests should always be conducted on a new install of RCTab prior to its use in an election.

Section 06 - System Design Specifications

System Design Specifications describe the configuration options available to users when installing the RCTab software and briefly describe voting and audit data retention requirements.

Section 07 - System Security Specification Requirements

The System Security Specification Requirements describe processes and tools necessary to ensure access control, equipment and data security, software installation and security, air gap, event logging, physical security, setup inspection, cryptography, telecommunications, and other elements of an effective security program when deploying RCTab.

Section 07I - Design and Interface Specification

This document provides a high-level design of RCTab, discusses external interfaces, and identifies threats RCTab protects against.

Section 07J - Security Architecture

This document provides an architecture level description of how the security requirements are met, and includes various authentication, access control, audit, confidentiality, integrity, and availability requirements.

Section 07K - Development Environment Specification

This document describes the physical, personnel, procedural, and technical security of the development environment including version control, tools used, coding standards used, software engineering model used, and a description of developer and independent testing

Section 07L - Security Threat Analysis

This document identifies the threats the voting system protects against and the implemented security controls on voting system and system components.

Section 07M - Security Testing and Vulnerability Analysis Documentation

This document describes security tests performed to identify vulnerabilities and the results of the testing. This also includes testing performed as part of software development, such as unit, module, and subsystem testing.

Section 09 - System Maintenance Manual

This section discusses the support needed to adjust or repair components of RCTab. RCTab leverages content from the jurisdictions voting system, all maintenance on equipment should be referred to your voting system vendor. All RCTab hardware is COTS and software other than RCTab are also COTS.

Section 10 - Personnel Deployment and Training

It is recommended that RCTab is used by at least two people in compliance with the jurisdictions' guidelines for partisan participation. Personnel should generally have a basic knowledge of desktop applications with some additional skills required for testing. Training requirements vary between two to eight hours depending on the task.

Section 11 - L\&A Testing

The Logic \& Accuracy Test document lays out an L\&A for RCTab that will allow jurisdictions to verify that RCTab is correctly configured and operating properly.

Section 12 - Configuration Management Plan

The Configuration Management Plan describes RCTab's development processes, how different versions of the software are managed, how to identify specific versions of the software, and provides details for functional and physical configuration audits of RCTab.

Section 13 - Quality Assurance Plan

This section describes the quality assurance plan used in RCTab development. It covers requirements, design process, and definition of RCTab software; determination of the specifications that a COTS device must meet in order to optimize RCTab installation and operation; process recommendations for centralization of CVR data prior to round-by-round counting; the validation and verification of the performance of RCTab; and 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.

Section 14 - Tabulator Trusted Build Instructions

This section explains how to create a trusted build of RCTab. It also provides a method for verifying whether the trusted build was successful.

Section 15 - System Change Notes

This section details the changes for each version of RCTab resulting from previous testing and certification. RCTab is currently used to produce official RCV results, as a testing tool, or as an auditing tool. The State of New York certified RCTab for use in single-winner RCV elections in the State, the State of Utah certified the RCTab for use in local RCV elections, and the State of Michigan certified RCTab for use in Eastpointe, Michigan's RCV elections.

Section 16 - System Hardening Procedures - Windows OS

The system hardening procedures describe the steps that should be taken to secure an RCTab workstation against various potential attacks on the system. It describes how to harden the OS, how to retrieve verifiable versions of software for use on the RCTab workstation, and procedures for locking down external ports on an RCTab workstation.

Section 17 - System Test and Verification Specification

The System Test and Verification Specification lays out all tests regularly conducted on the RCTab software and describes how to determine if a test of RCTab succeeds or fails. It also provides detailed information about each individual test condition used to test the software.

Section 18 - User Guide

The User Guide provides a step-by-step guide for using RCTab. It walks users through launching the software, creating an RCTab configuration file, generating results files, and securing any results files.

Section 19 - Tabulation Options for RCV Tabulation

The Tabulation Options section is an enumeration and discussion of the various tabulation options that exist for Ranked Choice Voting (RCV) elections and how those options are or are not incorporated into RCTab. It also includes a glossary of ranked choice voting terms.

Section 20 - Process Ranked Choice Voting Contest

This section consists of an introduction, a flowchart, and a description of the flowchart laying out how RCV contests should be processed according to various rules in place in jurisdictions in the United States.

Section 21 - Ballot Limitations & Maximum Testing Range

This document describes RCVRC's understanding of the maximum ballot ranges possible when creating RCV data from different voting system vendors.

Section 22 - Installation Instructions for Windows OS

This document describes the Windows OS installation process for RCTab.

Section 23 - Trusted Build & HashCode Verification - Windows OS

This document describes how to generate HashCodes when working on a Windows-based RCTab workstation.

Section 24 - Tabulator Command Line Instructions

This document describes how to launch and operate RCTab from the command line.

Section 25 - Configuration File Parameters

This document describes all parameters included in configuration files in the RCTab software.

Section 26 - RCTab CVR Files

This document describes the different CVR files RCTab is compatible with. It documents how they are laid out and describes relevant file structures that inform how RCTab parses CVR data.

Section 27 - RCTab Config Files

This document provides an example of an RCTab configuration file.

Section 28 - Post-Election Audit & Clearing RCTab from System

This document describes how to run a post-election tabulation audit and software audit of an RCTab installation. It also describes how to clear RCTab from a workstation if a fresh installation of RCTab is required.

Section 29 - RCTab Operator Log Messages

This section lays out all potential messages a user may receive through the RCTab operator log. It also suggests resolutions for any SEVERE errors that cause tabulation to fail.

Section 30 - RCTab System Tab Hints

This section recreates the Hints tab displayed to users in the RCTab UI.

Section 31 - Coding and Header Comment Standards for RCTab

This section lays out the coding and header comment standards used in developing RCTab.

Section 32 - Secure USB Process

This section outlines the details of how the RCVRC selects and secures a USB for use and transport to the requestee.

2.1.1 Appendices

Appendix I - Expected Outcome RCV Test Sets Multi-Winner

These tables lay out various tabulation conditions and expected outcomes in multi-winner RCV contests. They were used when developing the RCTab software to ensure different conditions were correctly handled by the software.

Appendix II - Expected Outcome RCV Test Sets Single-Winner

These tables lay out various tabulation conditions and expected outcomes in single-winner RCV contests. They were used when developing the RCTab software to ensure different conditions were correctly handled by the software.

Appendix III - Ranked Choice Voting Laws

This spreadsheet lays out the RCV statutes and regulations that governed development of the RCTab software.

Appendix IV - 22-Month Archiving Procedure

This document provides a procedure for archiving RCTab and any election results data from its use in elections.

2.2 Section 01 - System Overview

The Ranked Choice Voting Resource Center (RCVRC), a nonprofit organization, and Bright Spots, a software development team, have joined together to develop RCTab as an open-source software package that provides post-election tabulation to determine results in a ranked choice voting election.

RCTab can process data from voting machines that are capable of exporting cast vote records (CVRs) and tabulate a single-winner or multi-winner ranked choice voting election according to the rules used in current state, county, or city election jurisdictions in the United States.

RCTab is hosted and developed on GitHub. This allows individuals and teams to collaborate on the development of RCTab. GitHub allows developers to work on different parts of the codebase simultaneously and merge their changes seamlessly, all under the close supervision of the Ranked Choice Voting Resource Center and Bright Spots.

Users of RCTab should, in accordance with section RCTab Section 03 - System Hardware Specification, procure a Windows PC with the following specifications:

- Windows 10 or above Operating System
- 3.0 GHz processor
- 32GB RAM
- 10GB disk space

RCTab outputs the tabulation results in comma-separated values (.csv) tabular data format and creates an audit file for the RCV election. More information about the operation, software design, hardware requirements, and system design requirements of RCTab is available in the following sections:

- **Section 02 - Software Design and Specifications**
- **Section 03 - System Hardware Specification**
- **Section 06 - System Design Specifications**
- **Section 18 - User Guide**

2.3 Section 02 - Software Design and Specifications

2.3.1 Coding Standards and Style

We use Google Java Style Guide as our published, reviewed, and industry-accepted code style. For more details see page 93 of the VVSG Volume 1.0 guide.

Code development and review processes are described in **Section 13 - Quality Assurance Plan**.

Java 17

The tabulator is written in Java because it meets the VVSG and California 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 17.0.2 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 27 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 (17.0.2) which executes compiled object code when installed and run on the target system.

2.3.2 Tabulator Java classes

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

- `AuditableFile`: Create a file that, on close, is read-only and has its hash added to the audit log.
- `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 performs extensive validation to confirm that a config file contains parameters that are permitted by the software 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, a full Common Data Format JSON file, and corresponding .hash files. When "tabulate by precinct" is enabled, it also produces separate summary files for each precinct.
- `SecurityConfig`: Configuration for the cryptographic signing of a Hart file.
- `SecuritySignatureValidation`: A set of tools to verify signatures of Hart CVRs.
- `SecurityTests`: Test the cryptographic signature validation functionality.
- `SecurityXmlParsers`: In-memory representation of `.sig.xml` signature files.
- `StreamingCvrReader`: Contains the logic for parsing a CVR source file 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 a CVR file into a Common Data Format CVR 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.

2.3.3 3rd-party Modules

RCTab incorporates several 3rd-party modules which are all open-source. These meet the VVSG and California 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	1.9	CSV "Comma Separated Values" reader / writer.	https://commons.apache.org/proper/commons-csv/user-guide.html
Apache POI OOXML	5.2.2	Excel spreadsheet reader / writer.	https://poi.apache.org/apidocs/dev/org/apache/poi/ooxml/package-summary.html
Jackson Core	2.13.3	XML / JSON streaming reader / writer core	https://github.com/FasterXML/jackson-core
Jackson Annotations	2.13.3	XML / JSON deserialization annotations	https://github.com/FasterXML/jackson-annotations
Jackson Databind	2.13.3	XML / JSON deserialization	https://github.com/FasterXML/jackson-databind
Jackson Dataformat XML	2.13.3	XML reader / writer	https://github.com/FasterXML/jackson-dataformat-xml
Jupiter JUnit API	5.9.0	Automated testing (used only during development)	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
Jupiter JUnit Engine	5.9.0	Automated testing (used only during development)	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine
BouncyCastle FIPS API	1.0.2.4	RSA Validation	https://mvnrepository.com/artifact/org.bouncycastle/bc-fips/1.0.2.4

Software Limits: Limitations of the Tabulation software i.e., how many CVRs can be tabulated are detailed in **Section 03 - System Hardware Specification** document.

2.3.4 Contest Tabulation Logic

Overview

When the user triggers a contest tabulation, the application creates a new `TabulatorSession` object to manage the process flow. The `TabulatorSession` 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 **Section 18 - User Guide** and **Section 25 - Configuration File Parameters**. Note that this document covers all tabulation logic in RCTab in order to fully describe the software design. All tabulation logic is described in the 'Tabulation of CVRs' section below, 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 the manufacturer 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 tabulation 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:

a. 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.

b. 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.

c. If the rankings at this rank constitute an overvote, the CVR is handled according to the overvote rule set in the config.

d. If a continuing candidate is found at this rank, the CVR is marked as counting toward the candidate.

e. 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:

   `winningThreshold` = 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:

   `winningThreshold` = (`V` /(`N` + `1`) + 10^(-1 * `decimalPlacesForVoteArithmetic`) if `nonIntegerWinningThreshold` is set to `true`

   `winningThreshold` = floor(`V` /(`N` +1)) + 1 if `nonIntegerWinningThreshold` is set to `false`

   `winningThreshold` = floor(`V` / `N`) 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:

   `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_WINNER` S 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:

```
`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 `CastVoteRecord`s 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 - `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 19 - Tabulation Options for RCV Tabulation**.

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:

1. For a single-seat contest, one of these is true:

a. A winner has not been identified yet.

b. “Continue until two candidates remain” is enabled and one of these is true:

i. There are more than two candidates remaining.

ii. 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.)

2. For a multi-seat contest, one of these is true:

a. The winning mode is `MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD` and no winners have been identified yet.

b. 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.

c. The winning mode is neither `MULTI_SEAT_BOTTOMS_UP_USING_PERCENTAGE_THRESHOLD` nor `MULTI_SEAT_BOTTOMS_UP_UNTIL_N_WINNER`S 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 the following files with corresponding hashes 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

Each of these files contains a corresponding `fileName.hash` file. These files can be used to verify the content of their corresponding result files.

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:

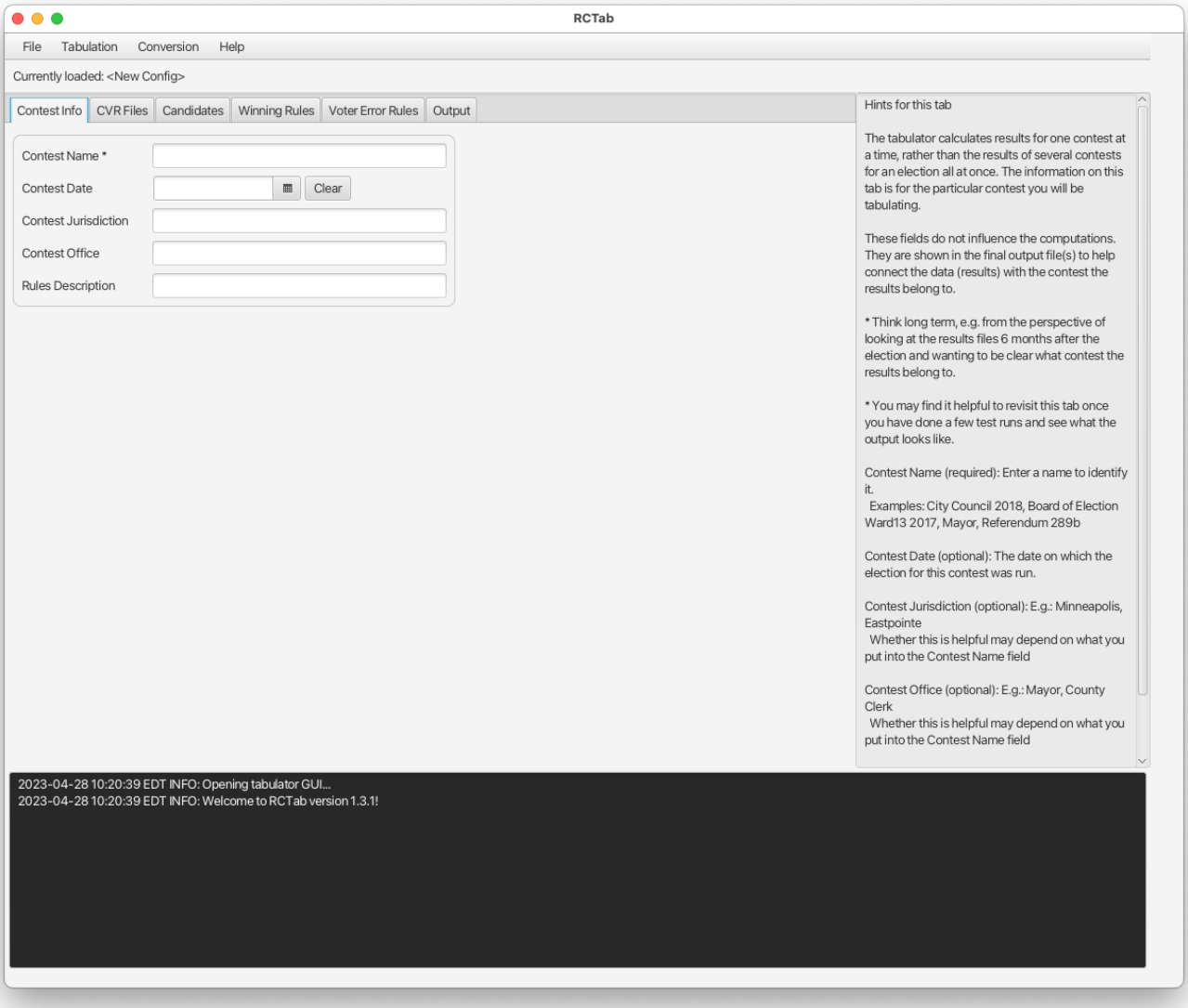
STREAM 1: TABULATOR "OPERATOR" LOGGING

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.

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.
Each audit_*N*.log is created with a corresponding audit_*N*.log.hash file. This hash file can be used to validate the contents of its corresponding audit_N.log file with the directions in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**

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 **Section 24 - Tabulator Command Line Instructions**. More information about error messages can be found in **Section 29 - RCTab Operator Log Messages**.

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 manufacturers (ES&S, Hart, CDF, Unisyn).

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

Section 07 - System Security Specification Requirements
Section 19 - Tabulation Options for RCV Tabulation
Section 20 - Process Ranked Choice Voting Contest
Section 21 - Ballot Limitations & Maximum Testing Range
Appendix III - Ranked Choice Voting Laws Appendix
Appendix II - Expected Outcome RCV Test Sets Single-Winner
Appendix I- Expected Outcome RCV Test Sets Multi-Winner

2.3.5 7.4.4 Software Distribution

b. The documentation shall designate all software files as static, semi-static or dynamic.

The `.json` config files used to configure RCTab are semi-static. These files are modified for contest-specific configuration options such as the candidate list, winning rules and output locations.

RCTab output files, if considered part of the “software files,” are also semi-static. The `summary.json`, `summary.csv`, `audit.log` and corresponding `.hash` files output for each tabulation are different depending on the configured contest.

All other RCTab files, excluding the `.json` config files and output files, are static. All remaining files’ contents (specifically the source code files and 3rd party dependencies) are not modified based on the election being conducted or the equipment upon which it is installed. All executable code files are the same no matter what hardware it is installed on and no matter what voting equipment it is being used with.

2.3.6 9.5.2 Applicable Documents

The manufacturer 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 the **Coding Standards and Style** paragraph of **Section 02 - Software Design and Specifications**.

2.3.7 9.5.3 Software Overview {#9.5.3-software-overview}

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

1. A description of the software system concept, including specific software design objectives, and the logic structure and algorithms used to accomplish these objectives.

- This is described in the **Contest Tabulation Logic** section of **Section 02 - Software Design and Specifications** and informed by the following documents:
- **Section 07 - System Security Specification Requirements**
- **Section 19 - Tabulation Options for RCV Tabulation**
- **Section 20 - Process Ranked Choice Voting Contest**
- **Section 21 - Ballot Limitations & Maximum Testing Range**
- **Appendix III - - Ranked Choice Voting Laws Appendix**
- **Appendix II - Expected Outcome RCV Test Sets Single-Winner**
- **Appendix I - Expected Outcome RCV Test Sets Multi-Winner**

2. The general design, operational considerations, and constraints influencing the design of the software.

- This is described in the **Contest Tabulation Logic** section of **Section 02 - Software Design and Specifications** and informed by the following documents:
- **Section 07 - System Security Specification Requirements**
- **Section 19 - Tabulation Options for RCV Tabulation**
- **Section 20 - Process Ranked Choice Voting Contest**
- **Section 21 - Ballot Limitations & Maximum Testing Range**
- **Appendix III - Ranked Choice Voting Laws Appendix**
- **Appendix II - Expected Outcome RCV Test Sets Single-Winner**
- **Appendix I - Expected Outcome RCV Test Sets Multi-Winner**

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

a. Written in-house.

- All in-house items written in-house are described in this doc under **Tabulator Java Classes** located in **Section 02 - Software Design and Specifications.**

b. Procured and not modified; and

- All third-party modules are not modified. They are described in **Section 02 - Software Design and Specifications** under **Java 17, Architecture, 3rd-party Modules**

c. Procured and modified including descriptions of the modifications to the software and to the default configuration options. The manufacturer 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. More information on verifying dependencies can be found in **Section 14 - Tabulator Trusted Build Instructions**

2.3.8 9.5.4 Software Standards and Conventions

The manufacturer shall provide information that can be used by the SOS and S-ATA to support software analysis and test design. The information shall address standards and conventions developed internally by the manufacturer as well as published industry

standards that have been applied by the manufacturer. The manufacturer shall provide information that addresses the following standards and conventions:

1. 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 **Section 13 - Quality Assurance Plan**)
- Software is submitted for VSTL testing
- Iterate with test lab
- Release

2. Software design standards, including internal manufacturer procedures.
3. Software specification standards, including internal manufacturer procedures.
4. Software coding standards, including internal manufacturer procedures.
5. Testing and verification standards, including internal manufacturer procedures, that can assist in determining the program's correctness and ACCEPT/REJECT criteria; and
6. 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 **Section 13 - Quality Assurance Plan** and **Section 17 - System Test & Verification**.

2.3.9 9.5.5 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 17. For more information see the above section titled **Java 17**.
- 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.

9.5.5.1 Hardware Environment and Constraints

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

1. The logic and arithmetic capability of the processor.
2. Memory read-write characteristics.
3. External memory device characteristics.
4. Peripheral device interface hardware.
5. Data input/output device protocols; and
6. 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 **Section 03 - System Hardware Specification** for more details.

9.5.5.2 Software Environment

The manufacturer 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 17.0.2 included with OpenJDK 17.0.2 (build 17.0.2+8).
- See also **Section 14 - Tabulator Trusted Build Instructions**

2.3.10 9.5.6 Software Functional Specification

The manufacturer 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:

- **Section 01 - System Overview**
- **Section 18 - User Guide**
- **Section 19 - Tabulation Options for RCV Tabulation**
- **Section 25 - Configuration File Parameters**

9.5.6.1 Configurations and Operating Modes

The manufacturer 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 manufacturer shall provide:
1. A definition of the inputs to the function or mode (with characteristics, tolerances or acceptable ranges, as applicable);
2. An explanation of how the inputs are processed; and
3. A definition of the outputs produced (again, with characteristics, tolerances, or acceptable ranges as applicable).

RCTab only processes ballot data once it has been exported from an election management system. It then produces a final report. For more information about the components of RCTab with regard to data input/counting and generating final count that are relevant here, please see:

- **Section 01 - System Overview**
- **Section 18 - User Guide**
- **Section 19 - Tabulation Options for RCV Tabulation**
- **Section 25 - Configuration File Parameters**

9.5.6.2 Software Functions

The manufacturer shall describe the software's capabilities or methods for detecting or handling:
1. Exception conditions.
2. System failures.
3. Data input/output errors.
4. 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 **Section 29 - RCTab Operator Log Messages**.
- System failures such as out of memory, disk input/output errors, etc. will be logged in these locations as well.
- Production of statistical ballot data.
- Tabulator results are described in **Section 18 - User Guide** under **Running A Tabulation**.
- Data quality assessment; and
- For data quality assessment, we only operate upon CVR data formatted according to the standards the manufacturers have defined for their CVRs. See election system manufacturer 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.
- Security monitoring and control.
- Jurisdiction practices and policies govern the security monitoring control when using the tabulator. We recommend a minimum of 2 trained employees operate the system together at all times.
- For detailed information about security monitoring and control issues, please see **Section 07 - System Security Specification Requirements**.
- 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.

2.3.11 9.5.7 Programming Specifications

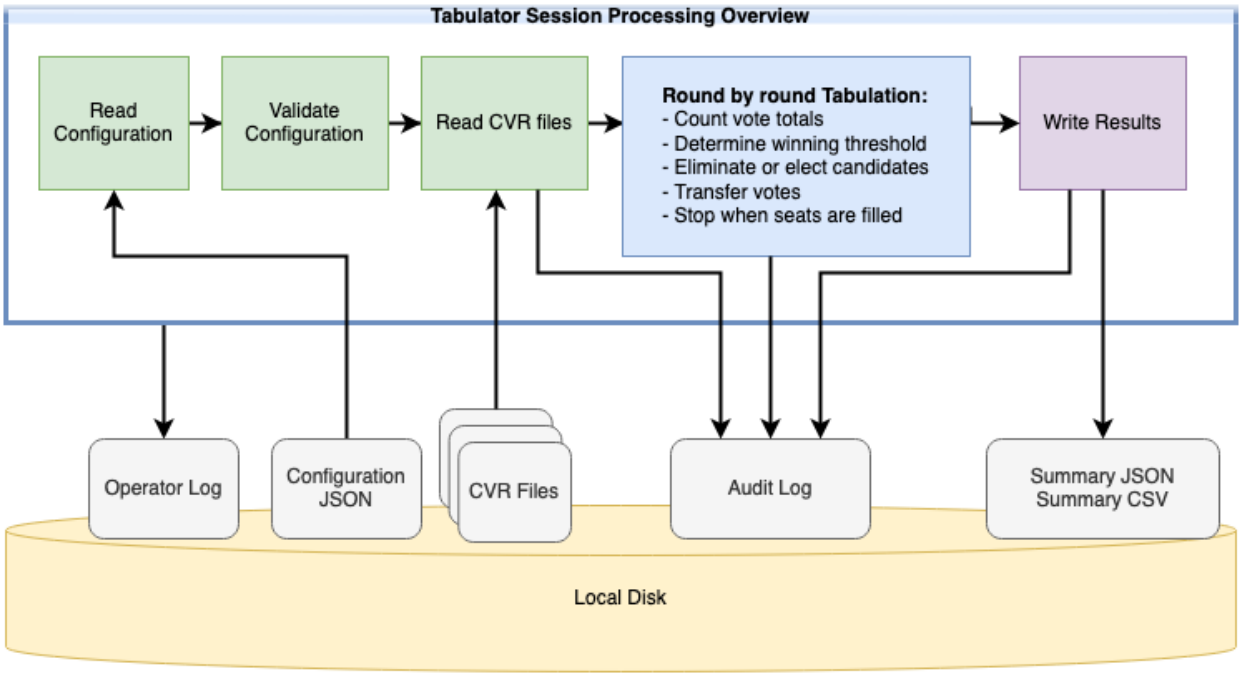
The manufacturer shall provide in this section an overview of the software design, its structure, and implementation algorithms and detailed specifications for individual software modules.

This is described in this document under **Contest Tabulation Logic.** Also see the below diagram located in 9.5.7.1.

9.5.7.1 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 the **Contest Tabulation Logic** section of **Section 02 - Software Design and Specifications.** Also see the below diagram:



9.5.7.2 Programming Specifications Details

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

1. Module and unit design decisions, if any, such as algorithms used.
2. Any constraints, limitations, or unusual features in the design of the software module or unit.
3. The programming language used and rationale for its use, if other than the specified module or unit language
4. If the software module or unit consists of, or contains, procedural commands (such as menu selections in a database management system for defining forms and reports, on-line queries for database access and manipulation, input to a graphical user interface builder for automated code generation, commands to the operating system, or shell scripts), a list of the procedural commands and reference to user manuals or other documents that explain them
5. If the software module or unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements as applicable.
6. If the software module or unit contains logic, the logic to be used by the software unit, including, as applicable:

a. Conditions in effect within the software module or unit when its execution is initiated.

b. Conditions under which control is passed to other software modules or units.

7. Exception and error handling

- The programming specifications that describe individual software modules and their component units are described in the **Tabulator Java Classes** and **3rd Party Modules** sections of this document.
- For additional information, see the following documents:
- **Section 01 - System Overview**
- **Section 18 - User Guide**
- **Section 19 - Tabulation Options for RCV Tabulation**
- **Section 25 - Configuration File Parameters**

8. If the software module is a database, provide the information described below in Subsection 9.5.8.

2.3.12 9.5.8 Databases

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

2.3.13 9.5.9 Interfaces

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

2.4 Section 03 - System Hardware Specification

Hardware requirements vary depending on the size (number of CVRs) of the contest you wish to tabulate. In general, the more RAM a system has the larger the elections it can run. The more powerful a computer's CPU is, the faster it will be able to process election results. Please note when tabulating large contests (more than 1 million CVRs) RCTab must be launched from the command line using the launch steps detailed below. In all cases, after running a tabulation, you must exit the tabulator to clear the memory. Launch RCTab again to tabulate another contest. No other applications should be running at the same time, as they may compete with the tabulator for resources (RAM and CPU) and cause it to run more slowly.

For a jurisdiction with a large number of registered voters, RCTab will need to configure a desktop computer that meets or exceeds the following specifications:

- Windows 10 operating system or above
- 3.0 GHz processor
- 32GB RAM
- 10GB disk space

The PC/Laptop is required to be set up according to the hardening procedures detailed in **Section 16 - System Hardening Procedures - Windows OS**

Please note when tabulating large contests (more than 1 million CVRs) RCTab must be launched from the command line using the launch steps detailed below.

2.4.1 Large Configuration:

- Goal: handle up to 6,000,000 CVRs containing up to 40 columns of rankings data each:
- 3.0 GHz processor
- 32GB RAM
- 10GB disk space + room for whatever log files your tabulations generate
- Produces ~6GB of audit log output per contest
- Runtime: less than 40 minutes
- Launch steps for a large election:
- Contest with more than 1,000,000 votes

a. Open a Command Prompt by navigating to the start menu and typing in Command Prompt.

b. Press enter to launch Command Prompt.

c. Change the current directory to the rcv folder created when you unzipped the Tabulator.

d. First, type in cd (note there should be a space after cd).

e. Using File Explorer navigate to the folder where RCTab is installed.

f. Double-click on the `rctab_v1.3.0_windows` folder

g. Click and drag the rcv folder over to the command prompt window

h. Your command prompt will now read something like:

i. `cd C:\Users\user\rctab_v1.3.0_windows\rctab_v1.3.0_windows\rcv`

i. Press enter

j. Launch the tabulator by entering the following command:

i. `.\bin\java -mx30G -p .\app -m network.brightspots.rcv/network.brightspots.rcv.Main`.

- This configuration will run a 6,000,000 record contest in about 30 minutes.

Additional smaller configurations are possible for smaller contests (see below)

2.4.2 Small configuration:

- Goal: handle up to 100,000 CVRs, up to 40 columns of rankings data each:
- 1 GHz processor
- 4GB of RAM
- 1GB of disk space + room for whatever log files your tabulations generate
- Produces ~100MB of audit log output per contest
- Runtime: less than 2 minutes
- To Launch: double-click the `rcv.bat` file
- Contests will be run in as long as one minute with this configuration. Smaller contests will run faster.

2.4.3 Medium Configuration:

- Goal: handle up to 1,000,000 CVRs, up to 40 columns of rankings data each:
- 3.0 GHz processor
- 16GB RAM
- 2GB disk space + room for whatever log files your tabulations generate
- Produces ~1GB of audit log output per contest
- Runtime: less than 5 minutes
- To Launch: double-click the `rcv.bat` file
- Contests will run in as long as 5 minutes with this configuration. Smaller contests will run faster.

Both small and medium contests can be run by using the same launch steps on the recommended computer. Contests of small and medium size can be run on the hardware defined in this document and in **Section 18 - User Guide** using the launch steps under the small and medium configurations.

2.4.4 Hardware Physical, Reliability, Maintainability, and Environmental Characteristics

Any machine used to run RCTab does not have any required physical characteristics beyond those expected of commercial off the shelf (COTS) computing equipment.

The RCTab software itself can be maintained by following **Section 09 - System Maintenance Manual** and following **Section 29 - RCTab Operator Log Messages**. These documents are written in a manner to be understood by non-technical election workers with sufficient training in the RCTab software. Any COTS hardware or software used to run RCTab should be maintained with reference to the Maintenance Procedures and to any relevant user guides and maintenance manuals included with that COTS equipment.

Any machine used to run RCTab should follow basic environmental conditions required of any COTS computing equipment.

2.5 Section 04 - System Functionality Description

RCTab is developed to run using the Java Development Kit (JDK) version 17.0.2. The Java platform includes an execution engine, a compiler, and a set of libraries used by RCTab. For more information, see **Section 02 - Software Design and Specifications**.

RCTab is designed to run as a stand-alone software compatible with Windows Operating System. Java 17 compatibility and system requirements are listed in the Java 17 Compatibility Document.

RCTab is normally received by the user election jurisdictions as a fully-compiled program in a Trusted Build by contacting the relevant authority directly for directions and procedures to receive the Trusted Build.

Git is the distributed version-control system used to track changes in the source code during software development. It is designed for coordinating work among programmers and is used to track changes in any set of the files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. Here is a link to additional Git information: Git External Documentation.

2.5.1 Hardware Functionality and Recovery Requirements

1. Restoration of the device to the operating condition existing immediately prior to an error or failure, without loss or corruption of voting data previously stored in the device

   RCTab relies upon exportable data from a jurisdiction's voting system. Data operated upon by RCTab may be stored on external devices such as flash drives or copied directly into the memory of the hardware RCTab software is installed on. If RCTab fails while tabulating results, the original files used should not be altered because the Tabulator does not directly operate upon CVR data. However, any loss of data caused by a failure of RCTab or the hardware upon which RCTab is installed should be rectified by exporting a new copy of the same data from the jurisdiction's voting system.

   If the hardware RCTab software is installed upon fails while tabulating results, a user should first restart the machine and RCTab and run the tabulation again. There is no persistent state of RCTab that would persist after a crash or after a machine restart.

   Original versions of voting data used in RCTab should be maintained on the jurisdiction's voting systems for as long as required by jurisdiction rules. See **Section 06 - System Design Specifications** and **Section 18 - User Guide** for more.

2. Resumption of normal operation following the correction of a failure in a memory component, or in a data processing component, including the central processing unit.

   If the hardware RCTab software is installed upon fails while tabulating results, a user should first restart the machine and RCTab and run the tabulation again. If the hardware continues to fail, users should consult the documentation included with the hardware to determine any error resolution steps. See **Section 09 - System Maintenance Manual** for more.

3. Recovery from any other external condition that causes equipment to become inoperable, provided that catastrophic electrical or mechanical damage due to external phenomena has not occurred

   If the hardware that RCTab is installed upon fails while tabulating results, a user should first restart the machine and RCTab and run the tabulation again. There is no persistent state of RCTab that would persist after a crash or after a machine restart. If failure persists, users should consult the documentation included with the hardware to determine any error resolution steps. See **Section 09 - System Maintenance Manual** for more.

2.5.2 Vote Tabulating Information

RCTab processes vote tabulation data exported from a jurisdiction's voting systems according to the rules as set up in the configuration file used by the user when operating RCTab. See **Section 02 - Software Design and Specifications** for more.

1. Monitor system status and generate machine-level audit reports

   RCTab provides users system status information through the log box in the user interface and through log documents (both tabulation audit logs produced after successful tabulation and the operator log file, which is available in the /bin folder on Windows computers). Audit logs capture details of successful tabulations, including all configuration file details, CVR parsing information, details on how every ballot was processed in every round of counting, and summary round-by-round results. The operator log captures all messages produced by RCTab: those produced during a failed tabulation, messages sent during a successful tabulation, messages sent while validating configuration files, and any other messages sent via the log box in the user interface. See **Section 02 - Software Design and Specifications** for more about audit reports and system status reports.

2. Accommodate device control functions performed by polling place officials and maintenance personnel

   Polling place officials will not interact with the RCTab software. Maintenance personnel should refer to **Section 09 - System Maintenance Manual** and **Section 18 - User Guide** when interacting with RCTab. All device control functions are laid out in those materials.

3. Register and accumulate votes

   RCTab processes digital cast vote records of ballots as cast by voters. Part of this requires the accumulation of cast vote record data; this is done in accordance with configuration requirements as set by users. See **Section 06 - System Design Specifications** and **Section 18 - User Guide** for more information about how RCTab handles cast vote records.

4. Accommodate variations in ballot counting logic

   RCTab relies upon cast-vote record data from a jurisdiction's voting system. CVR data must be adjudicated before being processed by RCTab. That cast-vote record data will have varying types of ballot data depending on how voters marked their ballots and how ballots were adjudicated, including candidate names, overvoted markings, skipped rankings, undervoted ballots, and write-in markings. RCTab has settings for users to determine how each of these different types of markings is processed by the software. RCTab performs no interpretation of voter marks on ballots, however, as CVR data used in RCTab software will include only non-ambiguous marks and data representing how voters marked their ballots. See **Section 25 - Configuration File Parameters** and **Section 18 - User Guide** for more information on the settings available to users for processing these varying voter marks.

   The only voting variation RCTab supports is ranked order voting. RCTab supports many permutations of ranked order voting (referred to as ranked choice voting in the documentation). See **Section 18 - User Guide** for more information on the permutations and settings available to users.

2.6 Section 05 - Acceptance Test Procedures

This Acceptance Test is designed to verify that RCTab is correctly configured and operating properly. Acceptance tests should always be conducted on a new install of RCTab prior to its use in an election. Also, acceptance tests should be conducted when there are significant changes or repairs made to the hardware where the software is installed. Example: If the RCTab computer is sent out for repair or if a new version of the operating system or RCTab is installed, an acceptance test should be conducted on RCTab as soon as it is returned.

2.6.1 Required Materials

In addition to the following, you must know the correct version of your operating system and RCTab. The manufacturer can provide assistance in determining the correct version. Users can also find that information in **Section 22 - Installation Instructions for Windows OS** 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 **Section 22 - Installation Instructions for Windows OS** document.
2. Recommended battery backup if the user jurisdiction facility does not have generator capabilities in the event of a power failure.

The RCTab computer system, consisting of items 1 & 2 above, should have been set up and tested for functionality 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. Requirements are included in **Section 03 - System Hardware Specification** and in **Section 16 - System Hardening Procedures - Windows OS**.

2.6.2 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.

2.6.3 Verify Correct Operating System and RCTab Software

Turn on the RCTab computer. 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. See **Section 03 - System Hardware Specification.** document and **Section 22 - Installation Instructions for Windows OS** document for procedures and more information.

Manufacturer recommendation: When your election management system successfully passes the acceptance test, affix to the top of the computer case a label with the date of the test and the initials of the person conducting the test. This information should also be recorded on the access log as per the user jurisdiction’s security guidelines and protocols.

2.7 Section 06 - System Design Specifications

5.1.1 - Configuration of software, both operating systems and applications, is critical to proper system functioning. Correct test design and sufficient test execution must account for the intended and proper configuration of all system components. Therefore, the vendors shall submit a record of all user selections made during software installation as part of the Technical Data Package. The vendor shall also submit a record of all configuration changes made to the software following its installation. The accredited test lab shall confirm the propriety and correctness of these user selections and configuration changes.

- RCTab software does not permit installation configuration changes by the user at any time during or after the installation process.
- RCTab software installation must be conducted according to **Section 22 - Installation Instructions for Windows OS**.
- See **Section 18 - User Guide** and **Section 16 - System Hardening Procedures - Windows OS** document for more information about RCTab operation and other software requirements on hardware used for RCTab software.

5.3.a - Maintain the integrity of voting and audit data during an election, and for at least 22 months thereafter, a time sufficient in which to resolve most contested elections and support other activities related to the reconstruction and investigation of a contested election.

- The user jurisdiction should follow all record retention policy requirements set by the jurisdiction itself and all other higher governing authorities with respect to the appropriate retention and storage of any materials generated by the software.
- The manufacturer recommends that in the absence of any user controlling records retention schedule, all materials generated by the software for a particular election be securely stored for a minimum period of 22 months.
- User must save all results files created for each contest in an election, including all configuration files used for those contests. This should also include testing data. User must export files to a hard drive secured and stored according to jurisdiction policies. For every contest processed through RCTab at least four files must be exported:
- Contest `configuration.json` file
- Contest summary `.csv` file
- Contest summary `.json` file
- Contest audit `.log` file(s)
- Depending on the size of the contest multiple `.log` files may be created. `.log` files have a maximum size of 50MB. Large contests frequently have many `.log` files.
- The **Section 18 - User Guide** suggests users create a folder for each contest where all relevant files for the contest are saved. If these procedures are followed, users will be able to copy whole sets of contest files over, folder by folder, on the external hard drive to be used for records retention.
- At the close of an election, users must also export any operator `.log` files, located in the bin folder of the RCTab installation folder. These `.log` files will be named `rcv_0.log`, `rcv_1.log`, and so on, depending upon how many `.log` files were produced during use of RCTab. These files must also be placed on the secured hard drive.
- After successfully copying over all relevant files to a secured hard drive, the hard drive should be stored in a temperature and humidity controlled area that meets the criteria for such media.

2.8 Section 07 - System Security Specification Requirements

2.8.1 9.6.1 Access Control

1. Manufacturers shall provide user and TDP documentation of access control capabilities of the voting system.

- **Section 16 - System Hardening Procedures - Windows OS** enumerates steps for creating a 'RCTab' Windows Standard user account on the RCTab machine. Installation instructions for RCTab in **Section 22 - Installation Instructions for Windows OS** describe how to set up and run the RCTab software using the 'RCTab' user account which has the absolute minimum permissions necessary. Following these instructions ensures the following:
- RCTab users cannot edit or delete RCTab summary output files or audit logs.
- RCTab users cannot edit or delete corresponding `.hash` files that can be used to verify the contents of all output files and audit logs.

2. Manufacturers shall provide descriptions and specifications of all access control mechanisms of the voting system including management capabilities of authentication, authorization, and passwords in the TDP.

- Access to RCTab should be at minimum made by no less than two employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training** for more.
- **Section 16 - System Hardening Procedures - Windows OS** enumerates steps for creating two Windows OS Accounts. One Administrator level Windows account for installation and initial configuration. A separate 'RCTab' Windows Standard user account with only necessary permissions that will not have access to make administrative changes to the machine that could be security issues.

  Installation instructions for RCTab in **Section 22 - Installation Instructions for Windows OS** describe how to properly run RCTab with minimum permissions necessary - being logged into the Windows OS as the 'RCTab' Windows standard user while using Window's 'Run As' capability to run the RCTab software **only** with Administrator privileges.

  This ensures that RCTab has the privileges necessary to ensure security (like setting output files to read-only) while simultaneously preventing the logged in human user from inheriting any of the same privileges or having any extra privileges not necessary for running a Tabulation.
- **Section 16 - System Hardening Procedures - Windows OS** describe setting a hardware BIOS password to protect the system BIOS settings from tampering. This locks down attempts to boot from unauthorized devices and prevents system configuration settings from being changed at a BIOS level.

3. Manufacturers shall provide descriptions and specifications of methods to prevent unauthorized access to the access control mechanisms of the voting system in the TDP.

- Access to RCTab should be at minimum made by no less than two employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training** for more.
- Access to the desktop or laptop should require password entry from the initial operating system for all users assigned to operate RCTab. Instructions in **Section 16 - System Hardening Procedures - Windows OS** and **Section 22 - Installation Instructions for Windows OS** provide all necessary steps so that RCTab users have the minimum permissions necessary to run and verify a contest tabulation.
- Jurisdictions shall restrict access to Windows Admin account and BIOS password to the minimum number of users necessary for initial setup. Election officials running the tabulation shall not have access to the Windows Admin account.

4. Manufacturers shall provide descriptions and specifications of all other voting system mechanisms that are dependent upon, support, and interface with access controls in the TDP.

- RCTab is hosted on a separate, standalone tabulation workstation. Users can bring flash drives holding cast vote record (CVR) data to and from the RCTab workstation to produce RCV results and bring RCV results back into the main voting system.

  The RCTab workstation does not directly interface, control, or support any other functions of the voting system.

5. Manufacturers shall provide a list of all of the operations possible on the voting system and list the default roles that have permission to perform each such operation as part of the TDP.

- After installation and set up following the instructions in **Section 16 - System Hardening Procedures - Windows OS** and **Section 22 - Installation Instructions for Windows OS** the only operations possible on an RCTab workstation are RCTab rules configuration, production of round-by-round RCV results, review of CVR data when setting up rules, review of audit logs and other event logs, and read-only review of round-by-round results after tabulation completes. All features are available to users once they have logged into the RCTab workstation.

9.6.1.1 General Access Control Policy

The manufacturer shall specify the features and capabilities of the access control policy recommended to purchasing jurisdictions to provide effective voting system security. The access control policy shall address the general features and capabilities and individual access privileges.

- Access to RCTab should be at minimum made by no less than two employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training** for more.
- Access to the desktop or laptop must require password entry from the initial operating system for all users assigned to operate RCTab. See **Section 16 - System Hardening Procedures - Windows OS** document for more information.
- Installation instructions for RCTab in **Section 22 - Installation Instructions for Windows OS** describe how to set up and run the RCTab software using the 'RCTab' Windows standard user account which has the absolute minimum permissions necessary to operate the software. Following these instructions ensures the following
- RCTab users cannot edit or delete RCTab summary output files or audit logs.
- RCTab users cannot edit or delete corresponding `.hash` files that can be used to verify the contents of all output files and audit logs.
- Employees accessing the software should be provided a paper log tracking for the purposes of establishing a recorded access record. The log should require the name of the personnel accessing the system, purpose of access, and the start and end times. This log should be maintained as part of the official election record and for as long as the user jurisdiction is required to maintain records per their controlling records retention schedule.
- The RCTab software provides full capabilities and access upon start-up and the manufacturer recommends not less than two user employees access the system simultaneously to prevent incorrect, accidental or on purpose, use of the features to tabulate ranked choice voting results.

SUGGESTED SOFTWARE, HARDWARE, AND PERSONNEL ACCESS CONTROLS

1. Suggested software access controls

- After successful completion of the steps to harden Windows OS and workstation hardware in **Section 16 - System Hardening Procedures - Windows OS** and steps to install the RCTab software in **Section 22 - Installation Instructions for Windows OS RCTab** the following software access controls apply to RCTab
- RCTab users cannot edit or delete RCTab summary output files or audit logs.
- RCTab users cannot edit or delete corresponding `.hash` files that can be used to verify the contents of all output files and audit logs.
- RCTab software will automatically, programmatically verify the cryptographic signature of the Hart CVRs used as input. If unable to verify, the software will not begin tabulation.
- The RCTab software employs a single user level, and the user has access to the capabilities of the software described in the **Section 25 - Configuration File Parameters** document and described by **Section 18 - User Guide**. The manufacturer recommends that at least two users visually observe the software during use.
- Access to RCTab should be at minimum no less than 2 employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training**.

2. Suggested hardware access controls

- Maintaining proper physical security to all computers is absolutely essential. When the equipment is in use, no less than two properly trained and trusted election officials should be present in the room with the equipment at all times. When the equipment is not being used (for instance, between elections), the computer and any backup hardware should be kept in a locked room, and entry to that room should be restricted and logged.

3. Suggested communications controls

- **Section 16 - System Hardening Procedures - Windows OS** requires users to turn off all networking features on hardware. No RCTab installed computer should ever run with its Wi-Fi enabled. No RCTab computer should ever be connected to any public network. For software installation, files should be captured to a USB flash drive using only secure methods. See also **Section 16 - System Hardening Procedures - Windows OS**

4. Suggestions for effective password management.

- Access to the desktop or laptop should require password entry from the initial operating system for all users assigned to operate RCTab. Any additional user jurisdiction security requirements should also be maintained when accessing any user jurisdiction owned equipment.

5. Suggestions for protection abilities of a particular operating system

- The user jurisdiction is fully responsible for maintaining owned hardware in keeping with universal maintenance and security standards including all updates and security patches for the operating system in use. See also **Section 16 - System Hardening Procedures - Windows OS** and **Section 09 - System Maintenance Manual**.
- Following the instructions in **Section 16 - System Hardening Procedures - Windows OS** and **Section 22 - Installation Instructions for Windows OS RCTab** enables the following protection abilities for the Windows Operating system
- Sets up an RCTab Windows standard user account with minimum permissions required to run RCTab
- Restricts OS folder permissions on the RCTab output folder to read only
- Enables Windows OS drive encryption
- Disables all OS-level network connections

6. Suggested general characteristics of supervisory access privileges

- Access to RCTab should be at minimum no less than 2 employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training**
- The user jurisdiction is fully responsible for assigning access to the software and procuring the suggested training (see **Section 10 - Personnel Deployment and Training**) for those employees.
- The user jurisdiction is also responsible for creating and maintaining a segregation of duties plan and a backup plan where there will be no less than two employees identified in the event that assigned personnel are unable to fulfill their assigned roles.

7. Suggested policies for segregation of duties

- Access to RCTab should be at minimum no less than 2 employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training**.
- The user jurisdiction is fully responsible for assigning access to the software and procuring the suggested training (see **Section 10 - Personnel Deployment and Training**) for those employees.
- The user jurisdiction is also responsible for creating and maintaining a segregation of duties plan and a backup plan where there will be no less than two employees identified in the event that assigned personnel are unable to fulfill their assigned roles.

8. Suggestions for any additional relevant characteristics

- There are no additional relevant characteristics associated with RCTab.

9.6.1.2 Access Control Measures

The manufacturer shall provide a detailed description of all system access control measures and mandatory procedures designed to permit access to system states in accordance with the access policy, and to prevent all other types of access to meet the specific requirements.

- RCTab uses two Windows OS accounts. The first is a Windows Administrator level account. Steps for its creation are described in **Section 16 - System Hardening Procedures - Windows OS**. RCTab shall be installed and initial configuration done by an election administrator with access to this account. **Section 22 - Installation Instructions for Windows OS** describes steps for the creation of a 'RCTab' Windows standard account. This new account does not have OS administrative privileges. Running a tabulation will be done only with the 'RCTab' standard user account. This ensures that once installation and initial configuration is complete, RCTab users will have minimum OS privileges necessary to run a tabulation and will not have access to make administrative changes to the machine that could be security issues. Election officials running the tabulation shall not have access to the Windows Admin account. The Windows Admin account credentials should be shared with the fewest number of users necessary to complete installation and initial configuration.
- Access to RCTab should be a minimum of two employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See **Section 10 - Personnel Deployment and Training**.

ADDITIONAL SUGGESTED ACCESS CONTROL MEASURES

1. Suggested measures for use of data and user authorization
   - Access to RCTab should be limited to no less than 2 employees within the user jurisdiction.
   - Upon startup, the software provides capabilities as described in **Section 25 - Configuration File Parameters** and **Section 18 - User Guide**. The manufacturer recommends that at least two users visually observe the software during use simultaneously to prevent incorrect, accidental or on purpose, use of the features to tabulate ranked choice voting results.
   - Access to RCTab should only be permitted during pre-election logic & accuracy testing and post-election tabulation of ranked choice voting results from the voting system generated cast vote records (CVR).
2. Suggested measures for program unit ownership and other regional boundaries
   - The user jurisdiction is fully responsible for assigning access to the software and procuring the suggested training (see **Section 10 - Personnel Deployment and Training**) for those employees.
   - The user jurisdiction is also responsible for creating and maintaining a backup plan in the event that assigned personnel are unable to fulfill their assigned roles.
3. Suggested measures for one-end or two-end port protection devices
   - RCTab software does not use any one-end or two-end port encryption devices.
4. Suggested measures for security kernels
   - RCTab software does not use security kernels.
5. Suggested measures for Computer-generated password keys
   - RCTab software does not use computer-generated password keys.
6. Suggested measures for special protocols
   - RCTab software does not require special protocols as data does not transmit in or out via a web-based system.
   - The manufacturer does recommend a dedicated USB flash drive and backup be secured with the hardware and used for no other purpose than software use.
7. Suggested measures for message encryption
   - RCTab does not require message encryption as data does not transmit in or out via a web-based system.

8. Suggested measures for controlled access security

- RCTab access takes place at a single central-count location. The software and hardware resides at this location. All election personnel are present at this location.
- Physical access to the site is controlled by policy and procedures under control of the jurisdiction.
- Physical access to the system hardware is controlled by policy and procedures under control of the jurisdiction.
- At no time should the hardware with installed software be connected to the internet via WiFi or ethernet.
- The user jurisdiction is fully responsible for assigning access to the software and procuring the suggested training (see **Section 10 - Personnel Deployment and Training**) for those employees.
- The user jurisdiction is also responsible for creating and maintaining a backup plan in the event that assigned personnel are unable to fulfill their assigned roles.

2.8.2 9.6.2 Equipment and Data Security

The manufacturer shall provide a detailed description of system capabilities and mandatory procedures for purchasing jurisdictions to prevent disruption of the voting process and corruption of voting data to meet the specific requirements. This information shall address measures for polling place security and central count location security.

- Access to RCTab should only be permitted during pre-election logic & accuracy testing and post-election tabulation of ranked choice voting results from the voting system generated cast vote records (CVR). RCTab should not be deployed in polling places. If the RCTab workstation is deployed and in use in a central count location, no less than two properly trained and trusted election officials should be present in the room with the equipment at all times. When the equipment is not being used (for instance, between elections), the computer and any backup hardware should be kept in a locked room, and entry to that room should be restricted and logged.
- Transportation of CVR data from the voting system to the RCTab workstation should follow jurisdiction procedures for handling election data. Any results data produced by RCTab should be secured following jurisdiction procedures.
- After successful completion of the steps in **Section 16 - System Hardening Procedures - Windows OS** and **Section 22 - Installation Instructions for Windows OS RCTab** RCTab contains the following capabilities to prevent corruption of voting data
- RCTab users cannot edit or delete RCTab summary output files or audit logs.
- RCTab users cannot edit or delete corresponding `.hash` files that can be used to verify the contents of all output files and audit logs.
- RCTab software will automatically, programmatically verify the cryptographic signature of the Hart CVRs used as input. If unable to verify, the software will not begin tabulation.

2.8.3 9.6.3 Software Installation and Security

1. The manufacturer shall provide a detailed description of the system capabilities and mandatory procedures for purchasing jurisdictions to ensure secure software (including firmware) installation to meet specific requirements. This information shall address software installation for all system components.

- User jurisdictions should install RCTab on a workstation configured according to the requirements laid out in **Section 16 - System Hardening Procedures - Windows OS**.
- RCTab installation should follow the steps laid out in **Section 22 - Installation Instructions for Windows OS** and **Section 05 - Acceptance Test Procedures**.

2. Manufacturers shall provide a list of all software related to the voting system in the technical data package (TDP).

- Software that must be installed on RCTab system:
- Windows 10 Pro, or above
- RCTab v1.3.2
- LibreOffice
- XML Notepad
- Users must also retain access to:
- Command Prompt
- Notepad
- Optional UPS
- See also **Section 16 - System Hardening Procedures - Windows OS**.
- No other software is necessary for user operation of RCTab.

3. Manufacturers shall provide at a minimum in the TDP the following information for each piece of software related to the voting system: software product name, software version number, software manufacturer name, software manufacturer contact information, type of software (application logic, border logic, third party logic, COTS software, or installation software), list of software documentation, component identifier(s) (such as filename(s)) of the software, type of software component (executable code, source code, or data).

- Documentation, manufacturer name, product name, version, certification application number of voting system, and file names and paths are all referred to with unique labels in documentation and in the system itself. See also **Section 16 - System Hardening Procedures - Windows OS** and **Section 13 - Quality Assurance Plan** document for information about version numbers and documentation numbers.

4. As part of the TDP, manufacturers shall provide the location (such as full path name or memory address) and storage device (such as type and part number of storage device) where each piece of software is installed on the voting system.

- RCTab does not require election specific programming to be created or installed. See also **Section 22 - Installation Instructions for Windows OS** and **Section 18 - User Guide**.

5. As part of the TDP, manufacturers shall document the functionality provided to the voting system by the installed software.

- All RCTab functionality is described in **Section 02 - Software Design and Specifications**, and **Section 18 - User Guide**.

6. As part of the TDP, manufacturers shall map the dependencies and interactions between software installed on the voting system.

- RCTab is designed to operate as a standalone piece of software on a tabulation workstation. RCTab is dependent upon the CVRs from the configured voting systems as input. RCTab does not depend on or interface with any other software installed on a voting system.
- Depending on the needs of user jurisdictions, users may use LibreOffice or NotePad to inspect CVRs when generating RCTab configuration files and may depend on Windows PowerShell or the Command Line to produce hash codes using the procedures in **Section 23 - HashCode Instructions - Windows OS**.
- There are no interactions between software installed on the system.

7. The manufacturer shall provide a detailed description of the system capabilities and mandatory procedures for purchasing jurisdictions used to provide protection against threats to third party products and services.

- Third party products and services installed on RCTab workstations should be configured according to the steps laid out in **Section 16 - System Hardening Procedures - Windows OS**.

9.6.3.1 Air Gap

The TDP for the voting system shall provide full procedures and instructions, to be incorporated into the Official Use Procedures for the voting system, to implement the segregated dual-installation architecture. Those procedures and instructions shall:

1. Require elections officials to use the permanent installation to lay out the ballot, define the election, and program all of the memory cards, including any DRE, ballot marking device, optical scan unit, etc.

- RCTab is not used to lay out ballots, define elections, or program memory cards.

2. Require elections officials to write a backup of the election database from the permanent installation onto write-once media (e. g., CD-R or DVD-R), carry the media by hand to the sacrificial installation, and install that database onto the sacrificial installation. After this point, the permanent installation shall not be used for the remainder of the election.

- RCTab is not used to set up election databases.

3. Require that, after the close of the polls, memory cards or other equipment containing votes returned from polling locations are uploaded to the sacrificial installation (not the permanent installation).

- RCTab does not interface directly with memory cards or other equipment after its return from polling locations.

4. Require that the sacrificial installation, not the permanent installation, is used to accumulate and tabulate election results, produce reports, and calculate the official election results.

- RCTab only interacts with cast vote records from sacrificial installations. It should therefore be used as part of any sacrificial installation.

5. Require that the "sacrificial" installation is treated as presumed-to-be-infected, so any machine or equipment that is ever connected to the sacrificial installation must never again be connected to the permanent installation.

- Results created by RCTab shall never be connected back to any permanent installation.

6. Ensure that any media that has been connected to the sacrificial installation is securely erased or reformatted before being used with the permanent installation.

- Any media connected to RCTab shall be erased or reformatted if it will be used with a permanent installation.

2.8.4 9.6.4 System Event Logging

1. Manufacturers shall provide TDP documentation of event logging capabilities of the voting devices.

- RCTab produces audit logs and tabulator operator logs. Logging capabilities are described in the RCTab Logging section of **Section 02 - Software Design and Specifications** and **Section 29 - RCTab Operator Log Messages**
- Windows OS also logs all events on the OS. Those event logs are available via the Windows Event Log application.

2. Manufacturers shall provide a technical data package that describes system event logging design and implementation.

- RCTab produces audit logs and tabulator operator logs. Log design and implementation are described in the RCTab Logging section of **Section 02 - Software Design and Specifications**. Detailed documentation describing how to read and make use of these logs is provided in **Section 28 - Post-Election Audit & Clearing RCTab from System**.

3. The technical data package shall provide the location (i.e. full path name or memory address) where each log is saved.

- RCTab saves event logs according to user settings as described in the RCTab logging section of **Section 02 - Software Design and Specifications** and the Generating Results Files section of **Section 18 - User Guide**.

2.8.5 9.6.5 Physical Security

1. Manufacturers shall provide a list of all voting system components to which access must be restricted and a description of the function of each said component.

- Maintaining proper physical security to all computers is absolutely essential. All RCTab workstations should have restricted physical access. When the RCTab workstation is in use, no less than two properly trained and trusted election officials should be present in the room with the equipment at all times. When the equipment is not being used (for instance, between elections), the computer and any backup hardware should be kept in a locked room, and entry to that room should be restricted and logged.

2. As part of the TDP, manufacturers shall provide a listing of all ports and access points of the voting system.

- Any RCTab workstation will have a varying set of ports depending upon the laptop or desktop on which RCTab is installed. Users must physically seal all external ports on hardware where RCTab is installed, 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.

3. For each physical lock used on a voting system, manufacturers shall document whether the lock was installed to secure an access point.

- Any RCTab workstation will have a varying set of ports depending upon the laptop or desktop on which RCTab is installed. Users must physically seal all external ports on hardware where RCTab is installed, 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.

4. Manufacturers shall provide a list of all physical security countermeasures that require power supplies.

- No RCTab physical security countermeasures require power supplies.

5. Manufacturers shall provide a technical data package that documents the design and implementation of all physical security controls for the voting system.

- Physical security controls and design are discussed in this section.

2.8.6 9.6.6 Setup Inspection

1. Manufacturers shall provide the technical specifications of how voting systems identify installed software in the TDP.

- RCTab relies on hash codes to verify that the correct version of the software has been installed. See **Section 05 - Acceptance Test Procedures**, **Section 22 - Installation Instructions for Windows OS** and **Section 23 - HashCode Instructions - Windows OS** for details.

2. Manufacturers shall provide a technical specification of how the integrity of software installed on the voting system is verified as part of the TDP. Software integrity verification techniques used to support the integrity verification of software installed on voting systems need to be able to detect the modification of software.

- RCTab relies on hash codes to verify that the correct version of the software has been installed. See **Section 05 - Acceptance Test Procedures**, **Section 22 - Installation Instructions for Windows OS**, and **Section 23 - HashCode Instructions - Windows OS** for details.

3. Manufacturers shall provide a technical specification of how the inspection of all the voting system registers and variables is implemented by the voting device in the TDP.

- RCTab does not have a voting device.

2.8.7 9.6.7 Cryptography

1. Manufacturers shall provide a list of all cryptographic algorithms and key sizes supported by the voting system.

- RSA Cipher with a 2048-bit key size

2. Manufacturers shall provide the technical specification of all cryptographic protocols supported by the voting system.

- We support validation of files encrypted using a detached Signature File, created by Hart Verity with .NET Framework 4.8.1. We do not sign any files, but only validate the signature of files signed by Hart Verity. Below, we outline exactly what that library does, and how we validate the files it produces.

  .NET Framework 4.8.1 creates a “Signature File” with the file extension .sig.xml. This Signature File includes data about both the signature itself, and about the file it is signing (the “Signed File”). The Signature File file includes the public key used to sign the file, the signature value, the file path to where the Signed File was originally signed, and the SHA-256 digest of the Signed File.

  From the Signature File, .NET Framework 4.8.1 constructs the following XML snippet, hashes it using SHA-256, then signs the hash of this snippet:

```
<SignedInfo xmlns="http://www.w3.org/2000/09/xmldsig#"><CanonicalizationMethod Algorithm="{CANONICALIZATION_URL}"></CanonicalizationMethod><SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"></SignatureMethod><Reference URI="file:///{FILE}"><DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"></DigestMethod><DigestValue>{SHA256_HASH}</DigestValue></Reference></SignedInfo>
```

- The following variables, above surrounded by curly brackets, are replaced with values specific to each signed file
- The value `{SHA256_HASH}` is the SHA-256 hash of the Signed File
- The value `{FILE}` is the filename and filepath of the Signed File
- The value `{CANONICALIZATION_URL}` is the URL used to uniquely identify the canonicalization algorithm
- To validate the signature, we require three files:
- The Public Key, which we store in a String in the source code to prevent runtime tampering, for which only one is valid for each release of RCTab.
- The Signed File
- The Signature File (`.sig.xml`) generated by .NET Framework 4.8.1
- We validate that the signature is valid using the following procedure:
- We verify that the Signed File matches the SHA-256 hash `{SHA256_HASH}` provided in the Signature File
- We verify that the Signed File’s filename matches the expected filename `{FILE}` found in the Signature File
- We verify that the public key found in the Signature File matches the expected Public Key that we have included in our source code
- We verify the cryptographic signature of the XML snippet above using a 2048-bit RSA with a 256-bit SHA hash.
- Additional notes:
- The original path of the signed file at the time of signature is included in the signed message. The path at the time of signature verification may be different - the files will have been moved from the Verity machine they were signed on to the RCTab machine for tabulation. The filename and a hash of the file contents, both of which are included in the signature itself, verify the contents of the signed file.
- We allow the `Algorithm` attribute of `CanonicalizationMethod` to vary, and support all CanonicalizationMethods supported by the `Java.Security` library in JDK 17.
- If the Signed File is an XML, it is not canonicalized before being hashed.
- The signature file IS canonicalized before being hashed.

3. Manufacturers shall provide the cryptographic module name, identification information (such as hardware/firmware/software name, model name, and revision/version number) and NIST FIPS 140-2 validation certificate number for all cryptographic modules that implement the cryptographic algorithms of the voting systems.

- RCTab uses #4616 BC-FJA (Bouncy Castle FIPS Java API), a Level 1 FIPS 140-2 Validated Cryptographic Module to verify the cryptographic signature of Hart Verity Signed CVRs.

  Note: We have BouncyCastle implemented as a provider, but we still use a non-FIPS-certified Sun implementation of a random number generator. The random number generator is only used to validate that the chosen public key is valid -- it's used as part of a primality-checking algorithm. Since we know that the public key we're given is already valid (as it is created by Hart Verity using their FIPS 140-2 certified implementation), this check is superfluous.

  We ensure that only FIPS 140-2 certified modules are used for the RSA algorithm by programmatically removing the non-certified providers before and re-adding them after.

4. Manufacturers shall map the cryptographic modules to the voting system functions the modules support. This requirement documents the actions of the voting system that invoke the cryptographic module.

RCTab Function	Module
Verify Hart CVR cryptographic signatures	Java.Security (JDK 17) BouncyCastle Provider (bc-fips) 1.0.2.3

5. When public key information is stored in a digital certificate (such as an X.509 certificate), manufacturers shall provide a description of all the certificate fields (such as names, algorithm, expiration date, etc.) including the default values for the voting system. If they exist, manufacturers shall provide any certificate policies associated with the digital certificate.

- The public key is stored in the Java source code. It has only two fields, the Exponent and the Modulus, which refer to the values in an RSA 2048-bit public key.

6. Manufacturers shall provide documentation describing how cryptographic keys are created, stored, imported/exported, and deleted by the voting system.

- Cryptographic public keys are provided via secure file transfer, and then included as part of the source code. As they are loaded, we use the Sun implementation of a random number generator to conduct a Miller-Rabin Test to ensure primality.

  We ensure that only FIPS 140-2 certified modules are used for the RSA algorithm by programmatically removing the non-certified providers before and re-adding them after.

2.8.8 9.6.8 Telecommunications and Data Transmission Security

The manufacturer shall provide a detailed description of the system capabilities and mandatory procedures for purchasing jurisdictions to ensure secure data transmission to meet specific requirements:

1. For all systems, this information shall address access control, and prevention of data interception

- Jurisdictions should follow their election data handling procedures to ensure prevention of data interception. See also **Section 16 - System Hardening Procedures - Windows OS**, the Access Control section of this document, and the Physical Security section of this document.

2.8.9 9.6.9 Other Elements of an Effective Security Program

The manufacturer shall provide a detailed description of the following additional procedures required for use by the purchasing jurisdiction:

1. Administrative and management controls for the voting system and election management, including access controls

- See sections 9.6.1 and 9.6.1.2 detailed description of access control in RCTab
- The user jurisdiction should assign and train employees as per **Section 10 - Personnel Deployment and Training** documentation. These employees should work in pairs whenever accessing the hardware where the software is installed and during actual use of the software itself.

2. Internal security procedures, including operating procedures for maintaining the security of the software for each system function and operating mode;

- The user jurisdiction should assign and train employees as per **Section 10 - Personnel Deployment and Training** documentation. These employees should work in pairs whenever accessing the hardware where the software is installed and during actual use of the software itself.
- The hardware containing installed software should be secured in a reasonably climate controlled area with access being permitted and monitored through the use of signed access logs and in compliance with any of the user jurisdiction’s own facility security policies.

3. Adherence to, and enforcement of, operational procedures (e.g., effective password management);

- The user jurisdiction should assign and train employees as per **Section 10 - Personnel Deployment and Training** documentation. These employees should work in pairs whenever accessing the hardware where the software is installed and during actual use of the software itself.
- The user jurisdiction should only install the software on hardware that is not ever connected to the internet via WiFi or ethernet connections. See also **Section 16 - System Hardening Procedures - Windows OS**.
- The hardware containing installed software should be secured in a reasonably climate controlled area with access being permitted and monitored through the use of signed access logs and in compliance with any of the user jurisdiction’s own facility security policies.

4. Physical facilities and arrangements; and

- The hardware containing installed software should be secured in a reasonably climate controlled area with access being permitted and monitored through the use of signed access logs and in compliance with any of the user jurisdiction’s own facility security policies.

5. Organizational responsibilities and personnel screening.

- The user jurisdiction should assign and train employees as per **Section 10 - Personnel Deployment and Training** documentation. These employees should work in pairs whenever accessing the hardware where the software is installed and during actual use of the software itself.
- The user jurisdiction is responsible for contacting the manufacturer to procure the recommended number of hours of training as per **Section 10 - Personnel Deployment and Training** documentation. This training may be done virtually or in person.

2.9 Section 07I - Design and Interface Specification

Objective: This document shall identify the threats the voting system protects against. This document shall provide a high-level design of the overall voting system and of each voting system component. It shall also describe external interfaces (programmatic, human, and network) provided by each of the computer components of the voting system (examples of components are DRE, Central Tabulator, Independent Audit machine).

RCTab is an open-source software package that provides post-election tabulation to determine results in a ranked choice voting election. RCTab can process data from voting machines that are capable of exporting cast vote records (CVRs) and tabulate a single-winner or multi-winner ranked choice voting election according to the rules used in current state, county, or city election jurisdictions in the United States.

The RCTab software is designed to tabulate results based on CVR data from voting system vendors such as ES&S and Hart InterCivic. RCTab does not rely on any network connections to produce results; it only needs compatible CVR data and a set of tabulation rules created by the user in the RCTab UI to produce RCV results.

2.9.1 System Security

When used with Hart Verity, RCTab 1.3.2 includes the following programmatic security mechanisms

- When tabulation begins, RCTab automatically, programmatically verifies the cryptographic signature of all Hart CVRs used as input for contest tabulation. This verification step ensures both the integrity (CVR contents have not been edited) and provenance (CVRs came from the Hart voting system) of the Hart CVRs. RCTab will throw a halting error and tabulation will not begin if cryptographic validation of Hart's CVR signature is not successful. This is to protect against tampering of CVRs between Hart Verity export and RCTab import by malicious internal actors.
- After successfully following installation instructions in **Section 22 - Installation Instructions for Windows OS** *only* the RCTab software is run with administrator privileges, the logged in Windows user does **not** have administrative privileges. All RCTab output files (audit logs, summary `.csv` and `.json`, corresponding `.hash` files) are programmatically set to Read-Only and cannot be edited by the RCTab Windows Standard user.

  This is to protect against any tampering with the content of the summary files or audit log by malicious internal actors. This also ensures that the tabulation user does not have admin-level OF privileges to make changes that could be security issues e.g. disabling read-only access or enabling network adaptors.
- Additionally, RCTab automatically, programmatically creates a cryptographic hash of all output files - audit logs, summary `.csv` and `.json` - that can be used to validate that those files have not been edited. This is to protect against any tampering with the content of the summary files and audit log.

The objective of these programmatic and procedural security improvements is to make the execution of internal malicious tampering so onerous that it will not be a viable target. The combination of procedural and new programmatic controls require significantly more time and complexity for any malicious actor - this makes RCTab a less appealing target. Additionally, these procedural and programmatic controls make auditing a tabulation to ensure secure, successful execution easier and quicker.

There are still security threats that exist. Certain programmatic requirements continue to require procedural steps. For example, properly setting read-only permissions on RCTab output folders and proper account credential security for the Windows Admin user account. By following the detailed installation instructions and initial configuration instructions in **Section 22 - Installation Instructions for Windows OS** and **Section 16 - System Hardening Procedures - Windows OS** these threats can be addressed.

RCTab is meant to be run on an air-gapped, non-internet-connected computer. This protects against any network-based attacks on the software. Any RCTab workstation should also be hardened against physical attacks by physically securing and/or shutting down access to any unnecessary ports on the RCTab workstation hardware. RCTab workstations should also be encrypted, physically secured under lock and key when not in use, and require a username and login to access the OS where RCTab is installed. This is to protect against a malicious internal actor who might attempt to connect to the internet to install malicious software/code.

RCTab relies on a human-computer interface to create tabulation rules configuration files and run any round-by-round tabulation. RCTab does not rely on any network or programmatic interfaces.

2.10 Section 07J - Security Architecture

Objective: This document shall provide an architecture level description of how the security requirements are met, and shall include the various authentication, access control, audit, confidentiality, integrity, and availability requirements.

Authentication: Access to RCTab should be at minimum, made by no less than two employees within the user jurisdiction. These employees should have received the suggested training time provided by the manufacturer before accessing the software. See also **Section 10 - Personnel Deployment and Training** Access to the desktop or laptop should require password entry from the initial operating system for all users assigned to operate RCTab. See also **Section 16 - System Hardening Procedures - Windows OS** document for more information.

Access Control: **Section 16 - System Hardening Procedures - Windows OS** enumerates steps for creating a 'RCTab' Windows Standard user account on the RCTab machine. Installation instructions for RCTab in **Section 22 - Installation Instructions for Windows OS** describe how to set up and run the RCTab software using the 'RCTab' user account which has the absolute minimum permissions necessary. Users should only access the software per jurisdiction approved rules and after users have obtained the recommended training as outlined in **Section 10 - Personnel Deployment and Training**. See also: **Section 07 - System Security Specification Requirements**.

Further description of the two Windows OS Accounts and they offer access control can be found in **Section 07 - System Security Specification Requirements** under headings 9.6.1 and 9.6.1.2.

Audit: RCTab produces audit logs and tabulator operator logs. Logging capabilities are described in the RCTab Logging section of **Section 02 - Software Design and Specifications**. Detailed documentation describing how to read and make use of these logs is provided in **Section 28 - Post-Election Audit & Clearing RCTab from System**. Windows OS also logs all events on the OS. Those event logs are available via the Windows Event Log application.

Confidentiality: While RCTab does use cast vote records to tabulate results, those records do not contain any voter-identifying information. Use of RCTab should only be performed by trained personnel. See also **Section 10 - Personnel Deployment and Training**.

Integrity: See **Section 03 - System Hardware Specification** for minimum operating specifications and **Section 16 - System Hardening Procedures - Windows OS** for procedures to ensure the hardware is adequately protected against unauthorized access, theft of data, and/or malicious attacks. Following any maintenance or replacement of equipment used to operate RCTab, users should refer to **Section 05 - Acceptance Test Procedures**. Integrity of CVR inputs as well as summary file and audit log outputs is addressed in **Section 07 - System Security Specification Requirements**

Availability: RCTab is used on COTS equipment. While equipment failure is rare, it should be recognized as a possibility. Jurisdiction backup and disaster plans should include strategies for handling equipment failures and replacements before they occur. See **Section 03 - System Hardware Specification** for minimum operating specifications and **Section 16 - System Hardening Procedures - Windows OS** for procedures to ensure the hardware is adequately protected against unauthorized access, theft of data, and/or malicious attacks. Following any maintenance or replacement of equipment used to operate RCTab, users should refer to **Section 05 - Acceptance Test Procedures**. The manufacturer also recommends conducting a post-installation and post-election hashing as outlined in **Section 23 - HashCode Instructions - Windows OS**.

2.11 Section 07K - Development Environment Specification

Objective: This document shall provide descriptions of the physical, personnel, procedural, and technical security of the development environment, including version control, tools used, coding standards used, software engineering model used, and description of developer and independent testing.

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

Coding Standards: We use *Google Checkstyle for Java* as our published, reviewed, and industry-accepted code style. For details, see Google Java Style Guide and page 93 of the VVSG Volume 1.0 guide.

Software Engineering Model: RCTab development uses the Waterfall model.

Description of Developer: The Ranked Choice Voting Resource Center (RCVRC), a nonprofit organization, and Bright Spots, a software development team, have joined together to develop RCTab as an open-source software package that provides post-election tabulation to determine results in a ranked choice voting election.

Independent Testing: We have developed a suite of 68 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. See **Section 17 - System Test and Verification Specification** for additional information.

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.

2.12 Section 07L - Security Threat Analysis

Objective: This document shall identify the threats the voting system protects against and the implemented security controls on voting system and system components.

2.12.1 Identified Threats & Mitigation

- Equipment Failure - RCTab is used on COTS equipment. While equipment failure is rare, it should be recognized as a possibility. Jurisdiction backup and disaster plans should include strategies for handling equipment failures and replacements before they occur.
- See **Section 03 - System Hardware Specification** for minimum operating specifications and **Section 16 - System Hardening Procedures - Windows OS** for procedures to ensure the hardware is adequately protected against unauthorized access, theft of data, and/or malicious attacks.
- Following any maintenance or replacement of equipment used to operate RCTab, users should refer to **Section 05 - Acceptance Test Procedures**.
- The manufacturer also recommends conducting a post-installation and post-election hashing as outlined in **Section 23 - HashCode Instructions - Windows OS**.
- If the user elects to use USB devices to move installation files and/or data to and from devices, the manufacturer recommends using instructions as outlined in **Section 32 - Secure USB Process**.

- Unauthorized Access - Software itself is vulnerable to attack if someone gains access to the tabulation computer/RCTab workstation without jurisdiction approval or knowledge because of violation of access controls
- Users should only access the software per jurisdiction approved rules and after users have obtained the recommended training as outlined in **Section 10 - Personnel Deployment and Training**. See also **Section 07 - System Security Specification Requirements**.
- When used with Hart Verity, RCTab 1.3.2 includes the following programmatic security mechanisms
- When tabulation begins, RCTab automatically, programmatically verifies the cryptographic signature of all Hart CVRs used as input for contest tabulation. This verification step ensures both the integrity (CVR contents have not been edited) and provenance (CVRs came from the Hart voting system) of the Hart CVRs. RCTab will throw a halting error and tabulation will not begin if cryptographic validation of Hart's CVR signature is not successful. This is to protect against tampering of CVRs between Hart Verity export and RCTab import by malicious internal actors.
- After successfully following installation instructions in **Section 22 - Installation Instructions for Windows OS** *only* the RCTab software is run with administrator privileges, the logged in Windows user does **not** have administrative privileges. All RCTab output files (audit logs, summary .csv and .json, corresponding .hash files) are programmatically set to Read-Only and cannot be edited by the RCTab Windows Standard user.

  This is to protect against any tampering with the content of the summary files or audit log by malicious internal actors. This also ensures that the tabulation user does not have admin-level OF privileges to make changes that could be security issues e.g. disabling read-only access or enabling network adaptors.
- Additionally, RCTab automatically, programmatically creates a cryptographic hash of all output files - audit logs, summary .csv and .json - that can be used to validate that those files have not been edited. This is to protect against any tampering with the content of the summary files and audit log.
- The objective of these programmatic and procedural security improvements is to make the execution of malicious tampering so onerous that it will not be a viable target. The combination of procedural and new programmatic controls require significantly more time and complexity for any malicious actor - this makes RCTab a less appealing target. Additionally, these procedural and programmatic controls make auditing a tabulation to ensure secure, successful execution easier and quicker.
- Though perfect security is the goal of RCTab, threats still exist. Certain programmatic requirements continue to require procedural steps. Not following the procedural steps could make it easier for a malicious actor to tamper with an election. For example, **Section 22 - Installation Instructions for Windows OS** describes explicitly setting permissions for the RCTab output folder to ensure read only access. If this is not done - either by not setting permissions during installation, or by configuring RCTab during tabulation to use a different folder - RCTab users cannot edit RCTab output but they can delete RCTab output. Someone could
- Copy the contents of the output
- Create a new file
- Edit the contents of that new file
- Delete the original file
- Then rename the edited file to the original

  But this is mitigated by the other security additions to v1.3.2. To successfully tamper with the `summary.csv` file for example, a malicious actor must also
- Rehash the edited file, delete the corresponding `.hash` file and recreate it with the new hash
- Delete the `summary.json` file, recreate it with the same edits to `summary.csv`, hash the `summary.json` file, delete it's corresponding `.hash` file and recreate it with the edited hash
- Find the `audit_X.log`, copy its contents to a new file, search through it to find and edit original hashes of the output files, rehash the `audit_X.log`, delete the corresponding `audit_X.log.hash`, and recreate it with the edited hash, delete the original `audit_X.log` and rename the edited one

  And all of these steps described above would be moot if installation instruction procedures are properly followed and permissions on the output folder set correctly. In that case, all RCTab output files are read only and cannot be edited, deleted, or recreated by the user running the tabulation.

- Incorrect assignment of RCV rules as set by the jurisdiction - Jurisdictions establish rules for interpreting and counting ranked choice voting results. Failure to correctly set up the RCTab configuration file according to jurisdiction requirements could lead to incorrect results.
- Pre-election testing should be performed as per jurisdiction policies and procedures. Instructions for conducting those tests can be found at **Section 11 - L&A Testing**

2.13 Section 07M - Security Testing and Vulnerability Analysis

Objective: This document shall describe security tests performed to identify vulnerabilities and the results of the testing. This also includes testing performed as part of software development, such as unit, module, and subsystem testing.

- We have developed a suite of 68 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. See **Section 17 - System Test and Verification Specification** for additional information.
- Previous VSTL tests with Pro V&V conducted security regression reviews of RCTab versions 1.0.1, 1.1.0, and 1.2.0, all of which incorporated the security policies of the baseline system. Those tests found RCTab to have an applied level of security compliant with the verified VVSG 1.0 security provisions. Testing also found that RCTab does not impact the ability of the baseline system as modified to satisfy the VVSG 1.0 security requirements.
- These tests also concluded that hash code procedures during the installation process and post-election process would help verify that the RCTab software as installed matches the trusted build of the software. See **Section 22 - Installation Instructions for Windows OS** and **Section 23 - HashCode Instructions - Windows OS** for more.
- A March 2021 test with SLI found that, “other than some high level access control assertions, RCVRC depends heavily on the security policies of the accompanying voting system, as well as the security policies of local jurisdictions.” Voting System Test Lab report results in early 2023 also found the need for enhanced security control. In response, version 1.3.2 was released with the following updates
- **Section 16 - System Hardening Procedures - Windows OS** enumerates steps for creating a ‘RCTab’ Windows Standard user account on the RCTab machine. Installation instructions for RCTab in **Section 22 - Installation Instructions for Windows OS** describe how to set up and run the RCTab software using the ‘RCTab’ user account which has the absolute minimum permissions necessary. Following these instructions ensures the following
- RCTab users cannot edit or delete RCTab summary output files or audit logs.
- RCTab users cannot edit or delete corresponding `.hash` files that can be used to verify the contents of all output files and audit logs.
- When tabulation begins, RCTab automatically, programmatically verifies the cryptographic signature of all Hart CVRs used as input for contest tabulation. This verification step ensures both the integrity (CVR contents have not been edited) and provenance (CVRs came from the Hart voting system) of the Hart CVRs. RCTab will throw a halting error and tabulation will not begin if cryptographic validation of Hart’s CVR signature is not successful.
- See **Section 07L - Security Threat Analysis** for additional analysis of potential threats to the RCTab software.

2.14 Section 09 - System Maintenance Manual

RCTab leverages content from the jurisdictions voting system, all maintenance on equipment should be referred to your voting system vendor. All RCTab hardware is COTS. All software other than RCTab software itself is also COTS. Refer to a COTS Maintenance Manuals for maintenance procedures.

Maintenance of RCTab software: Obtain an encrypted version of the trusted build from the governing body of your jurisdiction or the VSTL. Maintain that original copy in a secure, temperature- and humidity-controlled environment. Label it with the system version and hash value.

Recommended service actions to correct malfunctions or problems shall be discussed, along with personnel and expertise required to repair and maintain the system; and equipment, materials, and facilities needed for proper maintenance. This manual shall include the sections listed below.

2.14.1 9.9.1. Introduction

The manufacturer shall describe the structure and function of the equipment (and related software) for election preparation, programming, vote recording, tabulation, and reporting in sufficient detail to provide an overview of the system for maintenance and for identification of faulty hardware or software.

- Election preparation:
- The software can be used to set up jurisdiction configurations according to the ranked choice voting rules used in the jurisdiction’s RCV election(s). Tabulation rules options are menu selectable using the GUI. See also:
- **Section 11 - L&A Testing**
- **Section 18 - User Guide**
- **Section 25 - Configuration File Parameters**
- Programming
- RCTab is not used for programming elections
- Vote Recording
- RCTab processes RCV election data according to rules users input in configuration files. See also **[Section 18 - User Guide]**(user_guide.md).
- If users have issues with CVR files, refer to CVR procedures from jurisdiction and relevant procedures in **[Section 18 - User Guide]**(user_guide.md).
- Tabulation
- Tabulator is used solely with properly adjudicated CVR export files from the user jurisdiction and user jurisdiction’s primary voting system vendor.
- Reporting
- RCTab reports out summary results files in .csv and .json formats as well as an audit log in .log format. See also **Section 02 - Software Design and Specifications**.

The description shall include a concept of operations that fully describes such items as:

1. The electrical and mechanical functions of the equipment

a. RCTab runs using the power from the hardware referenced in **Section 03 - System Hardware Specification**.

2. How the processes of ballot handling and reading are performed (paper-based systems)

a. RCTab does not handle or read ballots.

3. How vote selection and casting of the ballot are performed (DRE systems);

a. RCTab does not handle vote selection or ballot casting.

4. How transmission of data over a network is performed (DRE systems, where applicable)

a. RCTab does not transmit data over a network.

5. How data are handled in the processor and memory units

a. Data are loaded from disk, stored in RAM, then processed in the CPU, periodically writing the data back to RAM and to disk.

6. How data output is initiated and controlled

a. In the output tab within RCTab, users can select the path for all RCTab output files. Output files will be .csv contest summary files, .json contest summary files, .log audit files and corresponding .hash files Optionally, users can configure RCTab to export .json CDF (common data format) files if “Generate a CDF JSON” is checked. See **Section 18 - User Guide** for additional information.

   When installed using the instructions in **Section 22 - Installation Instructions for Windows OS** RCTab output is read-only and cannot be modified or deleted. The contents of the output can also be verified cryptographically with their corresponding `.hash` using the instructions in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**

7. How power is converted or conditioned

a. Power comes from the hardware referenced in **Section 03 - System Hardware Specification**.

8. How test and diagnostic information is acquired and used

a. See **Section 11 - L&A Testing**

2.14.2 9.9.2 Maintenance Procedures

The manufacturer shall describe preventive and corrective maintenance procedures for hardware and software.

9.9.2.1 Preventative Maintenance Procedures

The manufacturer shall identify and describe:

1. All required and recommended preventive maintenance tasks, including software tasks such as software backup, database performance analysis, and database tuning

a. Confirm that you are using the most recent version of the software. Confirm this by reviewing the black log box at the bottom of RCTab, as seen in the screenshot below. The second line of text will read “Welcome to RCTab version [version number].” Confirm that this version number matches up with the version number required of your jurisdiction. This system’s number will be version 1.3.2.

b. All ballot adjudication must be completed prior to using CVR file(s) with RCTab. Such review must ensure the configuration file conforms to candidate names, undervote, overvote, and other labels used in CVR files. See also **Section 18 - User Guide**.

2. Number and skill levels of personnel required for each task

a. Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application.

3. Parts, supplies, special maintenance equipment, software tools, or other resources needed for maintenance

a. Recommend backup computer as well as backup USB stick stored in accordance with user jurisdiction security policies. See also:

- **Section 03 - System Hardware Specification**,
- **Section 16 - System Hardening Procedures - Windows OS**,
- **Section 22 - Installation Instructions for Windows OS**

4. Any maintenance tasks that must be coordinated with the manufacturer or a third party (such as coordination that may be needed for off-the-shelf items used in the system)

a. The manufacturer will confirm with user jurisdiction two months prior, or as time permits, to an election in which RCTab will be used that the user jurisdiction is using the most recently certified version of the software.

b. User jurisdiction must review relevant state and local laws regulating ranked choice voting elections. Ensure that configuration files produced in the use of RCTab conform to relevant laws. See **Section 18 - User Guide** for more.

c. User jurisdiction must review cast-vote record file formats from the voting system vendor providing CVRs for use with RCTab.

9.9.2.2.Corrective Maintenance Procedures

The manufacturer shall provide fault detection, fault isolation, correction procedures, and logic diagrams for all operational abnormalities identified by design analysis and operating experience. The manufacturer shall identify specific procedures to be used in diagnosing and correcting problems in the system hardware (or user-controlled software). Descriptions shall include:

1. Steps to replace failed or deficient equipment

a. Use **Section 29 - RCTab Operator Log Messages** and information provided by Operator Log Box to determine if the error has a known correction.

2. Steps to correct deficiencies or faulty operations in software

a. Use **Section 29 - RCTab Operator Log Messages** and information provided by Operator Log Box to determine if the error has a known correction.

3. Modifications that are necessary to coordinate any modified or upgraded software with other software modules

a. Any additional software should conform to the requirements described in this document and in **Section 16 - System Hardening Procedures - Windows OS**.

4. The number and skill levels of personnel needed to accomplish each procedure

a. Minimum of 2 (in compliance with the jurisdiction's guidelines/rules regarding partisan participation in tabulation functions). Skill level: knowledge of how to retrieve and install RCTab software from a trusted source (requires one-hour training).

5. Special maintenance equipment, parts, supplies, or other resources needed to accomplish each procedure

a. Recommend backup Tabulator computer as well as backup Tabulator USB stick stored in accordance with user jurisdiction security policies. See also:

- **Section 03 - System Hardware Specification**,
- **Section 16 - System Hardening Procedures Windows OS**,
- **Section 22 - Installation Instructions for Windows OS**

6. Any coordination required with the manufacturer, or other party, for off the shelf items

a. If no known correction, contact the manufacturer to review procedures already completed by the jurisdiction and propose resolution steps.

2.14.3 9.9.3 Maintenance Equipment

The manufacturer shall identify and describe a special purpose test or maintenance equipment recommended for fault isolation and diagnostic purposes.

- JUnit Jupiter 5.6.2 and JUnit Platform 1.6.2 are used for all automated testing of RCTab. See also **Section 03 - System Hardware Specification**.

2.14.4 9.9.4 Parts and Materials

Manufacturers shall provide detailed documentation of parts and materials needed to operate and maintain the system. Additional requirements apply for paper-based systems.

9.9.4.1. Common Standards

The manufacturer shall provide a complete list of approved parts and materials needed for maintenance.

- A computer conforming to the requirements laid out in **Section 03 - System Hardware Specification**.
- A backup USB drive with a copy of certified voting system software retrieved from a trusted source.

9.9.4.2 Paper-based Systems

RCTab is entirely software based. It does not rely upon any insertion of paper to produce results. This section does not apply to RCTab.

2.14.5 9.9.5 Maintenance Facilities and Support

The manufacturer shall identify all facilities, furnishings, fixtures, and utilities that will be required for equipment maintenance. In addition, manufacturers shall specify the assumptions made with regard to any parameters that impact the mean time to repair. These factors shall include at a minimum:

1. Recommended number and locations of spare devices or components to be kept on hand for repair purposes during periods of system operation

a. Recommend backup Tabulator computer as well as backup Tabulator USB stick stored in accordance with user jurisdiction security policies. See also:

- **Section 03 - System Hardware Specification**,
- **Section 16 - System Hardening Procedures - Windows OS**,
- **Section 22 - Installation Instructions for Windows OS**

2. Recommended number and locations of qualified maintenance personnel who need to be available to support repair calls during system operation

a. 1-2 qualified personnel. Personnel who have been adequately trained, as required above, may serve as maintenance personnel. Recommend at least one person available in or near the tabulation location. Manufacturer personnel can be made available for support and consultation both via phone/virtually and in-person.

3. Organizational affiliation (i.e., jurisdiction, manufacturer) of qualified maintenance personnel

a. Staff from the user jurisdiction should maintain the system.

2.14.6 9.9.6 Appendices

The manufacturer may provide descriptive material and data supplementing the various sections of the body of the System Maintenance Manual. The content and arrangement of appendices shall be at the discretion of the manufacturer. Topics recommended for amplification or treatment in the appendix include:

- **Glossary:** A listing and brief definition of all terms that may be unfamiliar to persons not trained in either voting systems or computer maintenance;
- See **Section 18 - User Guide** for all relevant terms used in RCTab.
- See **Section 25 - Configuration File Parameters** for details on all parameters in RCTab software.
- **References:** A list of references to all manufacturer documents and other sources related to maintenance of the system;
- **Section 03 - System Hardware Specification**
- **Section 18 - User Guide**
- **Section 16 - System Hardening Procedures - Windows OS**
- **Section 25 - Configuration File Parameters**
- **Detailed Examples:** Detailed scenarios that outline correct system responses to every conceivable faulty operator input; alternative procedures may be specified depending on the system state.
- **Section 29 - RCTab Operator Log Messages**
- **Maintenance and Security Procedures:** This appendix shall contain technical illustrations and schematic representations of electronic circuits unique to the system.
- No unique electronic circuits. No such illustrations are relevant.

2.15 Section 10 - Personnel Deployment and Training

The manufacturer shall describe the personnel resources and training required for a jurisdiction to operate and maintain the system.

2.15.1 9.10.1 Personnel

The manufacturer shall specify the number of personnel and skill levels required to perform each of the following functions:

1. Pre-election or election preparation functions (e.g., entering an election, race and candidate information; designing a ballot; generating pre-election reports;)
- Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application.
2. System operations for voting system functions performed at the polling place;
- No functions are performed at the polling place by this equipment.
3. System operations for voting system functions performed at the central count facility;
- Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application.
4. Preventive maintenance tasks;
- Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application.
5. Diagnosis of faulty hardware or software;
- Minimum of 2 (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: knowledge of how to retrieve and install RCTab software from a trusted source (requires one-hour training).
6. Corrective maintenance tasks; and
- Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application.
7. Testing to verify the correction of problems.
- Minimum of 2. (in compliance with the jurisdiction’s guidelines/rules regarding partisan participation in tabulation functions). Skill level: basic knowledge of how to interact with a desktop application and review ranked choice voting results.

A description shall be presented of which functions may be carried out by user personnel, and those that must be performed by manufacturer personnel.

- All functions can be performed by user personnel. Manufacturer personnel are not needed to perform any of the above functions. Manufacturer personnel are available to support as needed.

2.15.2 9.10.2 Training

The manufacturer shall specify requirements for the orientation and training of the following personnel:

1. Poll workers supporting polling place operations.
   - None. No functions are performed at the polling place by this equipment, so no poll workers will interact with the software.
2. System support personnel involved in election programming.
   - 2-6 hours of training provided by previously agreed-upon methods between manufacturer and user. Training may include but is not limited to video calls, in-person training, and tutorial videos.
3. User system maintenance technicians.
   - 2-6 hours of training provided by previously agreed-upon methods between manufacturer and user. Training may include but is not limited to video calls, in-person training, and tutorial videos.
4. Network/system administration personnel (if a network is used).
   - No network is used in the software.
5. Information Systems personnel; and
   - 2-6 hours of training provided by agreed-upon methods between manufacturer and user. Training may include but is not limited to video calls, in-person training, and tutorial videos.
6. Manufacturer personnel.
   - 4-8 hours of training. Training may include but is not limited to video calls, in-person training, and tutorial videos.

2.16 Section 11 - L&A Testing

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.

2.16.1 Required Materials

In addition to the following, you must know the correct version of your operating system and RCTab. The manufacturer can provide assistance in determining the correct version. Users can also find that information in **Section 22 - Installation Instructions for Windows OS** 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 **Section 22 - Installation Instructions for Windows OS** document.
2. A general use and directly connected printer (this includes necessary printer cable and power supply).
3. Recommended battery backup if the user jurisdiction facility does not have generator capabilities in the event of a power failure.
4. At minimum, 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.

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.

2.16.2 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.

2.16.3 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. See also **Section 03 - System Hardware Specification** document and **Section 22 - Installation Instructions for Windows OS** document for procedures and more information.

2.16.4 Logic & Accuracy Procedures

The procedures should be done with all assigned personnel in no less than teams of two who have received the recommended manufacturer-led training. All manufacturer-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 manufacturer recommends a name associated with the specific election (sample: "LA - RCV Election 11-5-2019"). The manufacturer 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 an 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 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 LibreOffice 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 **Section 29 - RCTab Operator Log Messages** 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 manufacturer 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 **Section 29 - RCTab Operator Log Messages** documentation and messages in the log box for how to resolve errors. If the error persists, restart the 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.

d. `.hash` files

i. Cryptographic hashes of the output files above

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 test 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 can verify result file hashes by following the instructions in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**

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 **Section 29 - RCTab Operator Log Messages**. Errors arising out of any hardware or software other than RCTab should refer to the **Section 09 - System Maintenance Manual** 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_portalnd_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 below. 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.

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" : { }
    } ]
    } ]
}
```

2.17 Section 12 - Configuration Management Plan

2.17.1 9.1 Configuration management plan

This document describes the Configuration Management Plan for the RCTab software.

2.17.2 9.2 Configuration management policy

RCTab v1.3.2 is a software-only utility. Tools and practices used to track development of RCTab are described below. In brief, we rely on the below practices to track and manage development of RCTab:

- Software tools for version control and update and issue tracking
- Configuration management policies, as outlined in this document
- Quality Assurance practices, as described in the QA Document and reflected in the GitHub issues tracking software

2.17.3 9.3 Configuration identification

RCTab consists of one non-COTS component: The RCTab software itself. A release of the RCTab software consists of the correctly identified version of the RCTab software, as identified following **Section 22 - Installation Instructions for Windows OS**.

Any RCTab workstation should include the correct version of the RCTab software. For this version, that is:

- RCTab v1.3.2

Any hardware used should conform to the requirements described in **Section 03 - System Hardware Specification**. Any additional software should conform to the requirements described in this document and in **Section 16 - System Hardening Procedures - Windows OS**.

9.3.1 and 9.3.2: Software Versioning and Naming conventions

Our versioning conventions follow the conventions for semantic versioning as described at https://semver.org/ and https://datasift.github.io/gitflow/Versioning.html. In brief:

Given a version number MAJOR.MINOR.PATCH-buildNo, we increment:

- **PATCH** when we fix something
- **MINOR** when we add a new feature
- **MAJOR** when we break backwards-compatibility or add major features
- We use the **buildNo** to differentiate different builds off the same branch, and to differentiate between development, test and production builds.

A major release significantly alters the functionality of RCTab and changes how the user interacts with RCTab.

A minor release makes small improvements, contains a collection of bug fixes, or otherwise adds functionality to RCTab without changing the user experience.

Patch releases fix any critical bugs that must be fixed before a user can use the RCTab software.

File naming conventions for files produced by RCTab are described in **Section 02 - Software Design and Specifications**.

DOCUMENTATION NAMING AND VERSIONING

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

[Software version] [Document Number-] [Document Name] [Document version]
Software Version: RCTab v1.3.2
Document version: 1.0.1
Example:
Section 11 - Quality Assurance Plan

Document names either reference the set of VVSG standards the document is responsive to or provide a brief description of the purpose or function of the information included in the document.

2.17.4 9.4 Baseline and promotion activities

The below sections describe how baseline software versions of RCTab are defined and how documentation baseline versions are defined.

Establishing a baseline for RCTab

RCVRC & Bright Spots define a baseline for the RCTab software when a software release version of RCTab is merged and tested following the configuration control procedures described below.

Promoting a subsequent version to baseline status & Maintenance Until Retirement

RCVRC & Bright Spots use an internal versioning scheme based upon the semantic versioning described above. Upon release of a version to the Voting System Test Lab, Bright Spots gives that version a three-part version number. If any bug fixes are needed in response to test findings, the third part of the version number (the PATCH label) is incremented to reflect the change.

The official version number is updated after a certification process completes.

Once a baseline is established, Bright Spots configures and incorporates all code changes according to the configuration control procedures described below. After code changes are regression tested and all tests pass, that version of the software is promoted to being the new baseline version of the software.

We use tags to track issues while they are being worked on, when any code updates are merged, and after issues are closed. Release notes for each release of RCTab note any changes made to the software and link out to the relevant issue for each change in the release. See https://github.com/BrightSpots/rcv/releases/ for examples of how these issues are tracked in release notes.

RCTab will be updated as necessary in the future, following the version control and source control procedures described in this document, until such time as the RCTab software itself is retired.

2.17.5 9.5 Configuration control procedures

9.5(a) Developing and Maintaining Internally Developed Items

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 RCTab and the reasoning behind them.

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 issues, 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.

Documentation was all drafted in Google Docs, which tracks changes to all documents through its built-in document history. However, our TDP documentation was written to address compliance issues and not as a proactive feature of development. We plan to conduct an audit of all documentation after certification and plan to create a system for tracking and updating

documentation into the future to ensure compliance with all requirements. Documents also include a Document Revision History box at the bottom of each document which notes dates of revision, describes any revisions made, tracks authors, and provides document version numbers. Document versioning will follow the versioning described above.

9.5(b) Acquiring and maintaining third party items

Third-party items included in the RCTab submission can be divided into two categories:

- COTS hardware that runs the RCTab software
- Open-source software artifacts included in the RCTab software

Hardware

RCTab software runs on COTS computers. Jurisdictions purchase this COTS hardware directly, based on requirements provided by RCVRC and Bright Spots. The hardware used for RCTab software is discussed in **Section 03 - System Hardware Specification**.

Software

All open-source software libraries and other tools included with RCTab software can be obtained and updated using the internet and are maintained by their respective authors unless stated otherwise in that product's documentation.

Bright Spots downloads the versions it requires and tracks the software versions needed for the operation of RCTab.

All third-party software included in RCTab software is unmodified.

Trusted Build instructions and verification steps are available in **Section 14 - Tabulator Trusted Build Instructions**.

The document **Section 02 - Software Design and Specifications** maintains a record of the software libraries and artifacts used in the RCTab software. For details on the software tools used to create RCTab, see **Section 02 - Software Design and Specifications**.

Additional COTS Software

Any RCTab workstation should also include the following COTS software:

- Windows 10 Pro, or above
- LibreOffice
- XML Notepad
- Users must also retain access to:
- Command Prompt
- Notepad
- UPS

All COTS software should be obtained from the original provider, as described in **Section 16 - System Hardening Procedures - Windows OS**.

9.5(c) & (d) Resolving internally and externally identified defects

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; and
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.

2.17.6 9.6 Release process

This section describes how RCTab is released to test labs and end users.

9.6.1 First release to an accredited test lab

RCVRC submits all RCTab artifacts necessary for testing to state election officials, or the relevant election officials. Those officials then submit materials to the test lab.

When RCTab is ready to download, RCVRC & Bright Spots sends an email to state election officials or test lab representatives. That email will contain a link to a downloadable version of the most recent build of RCTab and a SHA-512 hash code for the RCTab build shared. Downloads will either be provided through a Google Drive link or via the RCTab releases page on GitHub (https://github.com/brightspots/rcv). This SHA-512 hash code can be used to validate the version of the RCTab downloaded using the download link by following the instructions in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**.

After downloading and verifying files, the test lab will install the software as described in **Section 22 - Installation Instructions for Windows OS**.

9.6.2 Upgrade Release Scenario

In the event that any upgrades or changes are made to the RCTab system, the RCVRC & Bright Spots will send a new download link as described above to download a new version of RCTab. Any previous versions of RCTab on hardware being used by the test lab will need to be uninstalled, which can be done by deleting RCTab folder and any files created or used by RCTab (such as configuration .json files, summary result .csv and .json files, audit .log files, and any CVRs used with RCTab).

9.6.3 & 9.6.5 Delivery and Installation of System to Jurisdiction

Jurisdictions will obtain the Trusted Build version of RCTab from the relevant State Election Officials on a flash drive or some other form of digital media. **Section 22 - Installation Instructions for Windows OS** explains how to install and verify RCTab. The RCTab version installed is also included in the operator log box at the bottom of the RCTab user interface.

9.6.4 Maintenance or Upgrade Release of System to Jurisdiction

RCTab upgrades will be released to jurisdictions after they have gone through necessary testing by state officials and test labs. Releases will be provided to jurisdictions as described in 9.6.3.

2.17.7 9.8 Configuration management resources

RCTab v1.3.2 is a software-only utility. The manufacturer uses Git as its distributed version-control system to track changes in the source code during software development. It is designed for coordinating work among programmers and is used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. Here is a link to additional Git information: Git External Documentation (https://git-scm.com/doc/ext).

Git may be downloaded free via this link: Git Download (https://git-scm.com/). The Bright Spots developers use GitVersion as the tool to achieve *Semantic Versioning* on the RCTab software development and upgrades. Additional information is available in **Section 13 - Quality Assurance Plan** and here: GitVersion Information (https://datasift.github.io/gitflow/Versioning.html).

2.17.8 9.11 Configuration audits

The guidelines require two kinds of configuration audits:

- Physical Configuration Audits (PCA)
- Functional Configuration Audits (FCA)

9.11.1 Physical Configuration Audit

The Physical Configuration Audit is conducted by the S-ATA to compare the voting system components submitted for certification to the manufacturer's technical documentation.

For the PCA, a manufacturer shall provide:

9.11.1(A) IDENTIFICATION OF ALL ITEMS THAT ARE TO BE A PART OF THE SOFTWARE RELEASE

RCTab Software

Any RCTab workstation should include the correct version of the RCTab software. For this version, that is:

- RCTab v1.3.2

COTS Software

Any RCTab workstation should also include the following COTS software:

- Windows 10 Pro, or above
- LibreOffice
- XML Notepad
- Users must also retain access to:
- Command Prompt
- Notepad
- UPS

All COTS software should be obtained from the original provider, as described in **Section 16 - System Hardening Procedures - Windows OS**.

RCTab Documentation

For a list of documentation, see **Section 00 - Table of Contents**. Documentation is also provided on a flash drive along with the installation .zip for RCTab.

Acceptance test procedures and criteria

Acceptance test procedures and criteria are included in the following section:

- **Section 05 - Acceptance Test Procedures**
- **Section 11 - L&A Testing**
- **Section 17 - System Test & Verification Specification**
- **Section 22 - Installation Instructions for Windows OS**

There are no changes between the system used for the PCA and the system used for the FCA.

Materials needed for a configuration audit:

Software

Software	Description	SHA 512 Hash Value
RCTab v1.3.2	RCTab Software	See **Section 14 - Tabulator Trusted Build Instructions** for instructions on generating a SHA512 to validate RCTab
Any RCTab workstation should also include the following COTS software:		
LibreOffice 7.5.7	For viewing formatted .csv output	`30dd3e7f158d3a6289d474d8d8a90eb8995ff5847aab0b23dd4dcc6455e36374`
UPS		
Users must also retain access to or be granted access to built-in Windows 10, or above, components: • Notepad • Command Prompt		All COTS software should be obtained from the original provider, as described in **Section 16 - System Hardening Procedures - Windows OS**

9.11.1(B) SPECIFICATION OF COMPILER (OR CHOICE OF COMPILERS) TO BE USED TO GENERATE EXECUTABLE PROGRAMS

1. See **Section 03- System Hardware Specification** for additional hardware requirements.

9.11.1(C) IDENTIFICATION OF ALL HARDWARE THAT INTERFACES WITH THE SOFTWARE

1. See **Section 03- System Hardware Specification** for additional hardware requirements.

9.11.1(D) CONFIGURATION BASELINE DATA FOR ALL HARDWARE THAT IS UNIQUE TO THE SYSTEM

RCTab Hardware

RCTab relies on COTS hardware. See also **Section 03 - System Hardware Specification** for a complete list.

Hardware

Component Name.	Version Number	Operating System	See RCTab Section 03 - System Hardware Specification for additional hardware requirements.
RCTab Workstation	HP Z4 G4 workstation	Windows 10 Pro or above	

Peripherals

Part Name	Model Number	Description
Printer	Samsung Xpress M2020W Printer (or similar)	RCV Printer
Uninterruptible Power Supply	APC Backup UPS 600 (or similar)	RCV External Power Supply

Test Support Equipment/Materials

Component Name	Quantity	Description
8.5 x 11 Printer Paper	As needed	HP 200350
Flash drive	2+	For transporting cast-vote records and files produced by RCTab

9.11.1(E) COPIES OF ALL SOFTWARE DOCUMENTATION INTENDED FOR DISTRIBUTION TO USERS, INCLUDING PROGRAM LISTINGS, SPECIFICATIONS, OPERATIONS MANUAL, VOTER MANUAL, AND MAINTENANCE MANUAL

1. See **Section 18 - User Guide**,

9.11.1(F) USER ACCEPTANCE TEST PROCEDURES AND ACCEPTANCE CRITERIA

1. See **Section 05 - Acceptance Test Procedures**

9.11.1(G) IDENTIFICATION OF ANY CHANGES BETWEEN THE PHYSICAL CONFIGURATION OF THE SYSTEM SUBMITTED FOR THE PCA AND THAT SUBMITTED FOR THE FCA, WITH A CERTIFICATION THAT ANY DIFFERENCES DO NOT DEGRADE THE FUNCTIONAL CHARACTERISTICS

1. See also:

- **Section 09 - System Maintenance Manual**
- **Section 15 - System Change Notes**
- **Section 18 - User Guide**

9.11.1(H) COMPLETE DESCRIPTIONS OF ITS PROCEDURES AND RELATED CONVENTIONS USED TO SUPPORT THIS AUDIT BY:

1. Establishing a configuration baseline of the software and hardware to be tested

a. One Computer with RCTab installed. Installation instructions can be found in **Section 22 - Installation Instructions for Windows OS** document.

b. Recommended battery backup if the user jurisdiction facility does not have generator capabilities in the event of a power failure.

c. 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.

d. Turn on the RCTab computer. 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.

2. Confirming whether the system documentation matches the corresponding system components

a. Documentation is broken down into different sections to make identification of a specific procedure easier for the user. Documentation uses language directly from the RCTab software that the user would see as well as including images where applicable. Additionally, the **RCTab Section 18 - User Guide** provides step-by-step instructions for the user with reference to other sections for additional details as needed.

9.11.2 Functional Configuration Audit

The Functional Configuration Audit is conducted by the S-ATA to verify that the system performs all the functions described in the system documentation. The manufacturer shall:

1. Completely describe its procedures and related conventions used to support this audit for all system components
2. Provide the following information to support this audit:

a. Copies of all procedures used for module or unit testing, integration testing, and system testing

b. Copies of all test cases generated for each module and integration test, and sample ballot formats or other test cases used for system tests

c. Records of all tests performed by the procedures listed above, including error corrections and retests

All test cases for RCTab are available at https://github.com/BrightSpots/rcv/tree/master/src/test/resources/network/brightspots/rcv/test_data. Test and Verification requirements are also described in:

- **Section 05 - Acceptance Test Procedures**
- **Section 11 - L&A Testing**
- **Section 17 - Test & Verification Specifications**

2.18 Section 13 - Quality Assurance Plan

2.18.1 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.

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 uses CVR data exports from election tabulation equipment.

Items Covered in the Scope

- Requirements, design process, and definition of 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.
- The validation and verification of the performance of 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.

2.18.2 Requirements, Design Process, and Definition of RCTab Software

Requirements

RCTab is tabulation software that is designed to use voting system data (such as the Cast Vote Record (CVR) from a tabulation system) 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 **Section 03 - System Hardware Specification**.

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 the Ranked Choice Voting Resource Center or State Agency.

COTS computer that is not connected to the internet and is configured according to all specifications laid out in the following documents:

- **Section 03 - System Hardware Specification**
- **Section 07 - System Security Specification Requirements**
- **Section 12 - Configuration Management Plan**
- **Section 16 - System Hardening Procedures - Windows OS**

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

2.18.3 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.

Section 04 - System Functionality Description 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 **Section 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. See also **Section 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:** See **Section 17 - System Test and Verification Specification** for detailed information on the 68 RCTab 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, manufacturer 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:

a. reproducing the defect

b. making necessary code changes to fix the defect on an isolated git branch; requesting a code review and implementing any changes arising from the code review; and verifying that regression tests pass

c. pushing the code change to the main branch and linking the issue in the commit message

d. 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

The full suite of programmatic tests that are run with every update to RCTab are located in **Section 17 - System Test and Verification Specification**.

Product Documentation

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

[System Version][Document/Section Number] - [Document Name] [Document version]
System Version: RCTab v1.3.2

Document version: 1.0.0
Example: Section 13 - Quality Assurance Plan

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:

- **Section 01 - System Overview**
- **Section 02 - Software Design and Specifications**
- **Section 03 - System Hardware Specification**
- **Section 04 - System Functionality Description**
- **Section 07 - System Security Specification Requirements**
- **Section 09 - System Maintenance Manual**
- **Section 10 - Personnel Deployment and Training**
- **Section 12 - Configuration Management Plan**
- **Section 13 - Quality Assurance Plan**
- **Section 15 - System Change Notes**
- **Section 16 - System Hardening Procedures - Windows OS**
- **Section 17 - System Test and Verification Specification**
- **Section 18 - User Guide**

2.19 Section 14 - Tabulator Trusted Build Instructions

2.19.1 Environment variables

Other than where specifically indicated below, building the application and generating hashes does not require that the system have any environment variables defined in advance.

2.19.2 Full build

RCTab can be built on any COTS x64 Windows machine with

- Windows 10 or greater
- At least 1GB of free space
- 4GB of RAM

To generate a build from the source code and then generate a hash of that build, use the following steps.

There are 4 setup steps that must be followed before you can build RCTab. Those steps are

1. Get the source code
2. Prepare Java
3. Prepare Gradle
4. Get third party dependencies

SOURCE CODE

First, you need a local copy of the source code. You can either get it from the RCTab Release page or directly from Git.

RCTab Release Page

Navigate to the RCTab Release page. Find your version. For v1.3.2 in California it is v1.3.2 (`84df706`). Download and extract the Source code (zip).

Git

Important note for Windows users: to ensure that the files in your copy of the repository exactly match the ones that were originally used to generate the Windows release build, you must have Git's autocrlf setting enabled before you clone the repository. This setting will cause Git to automatically convert the line endings in each source file from Unix-style (`\n`) to DOS-style (`\r\n`) when it clones the repository to your machine.

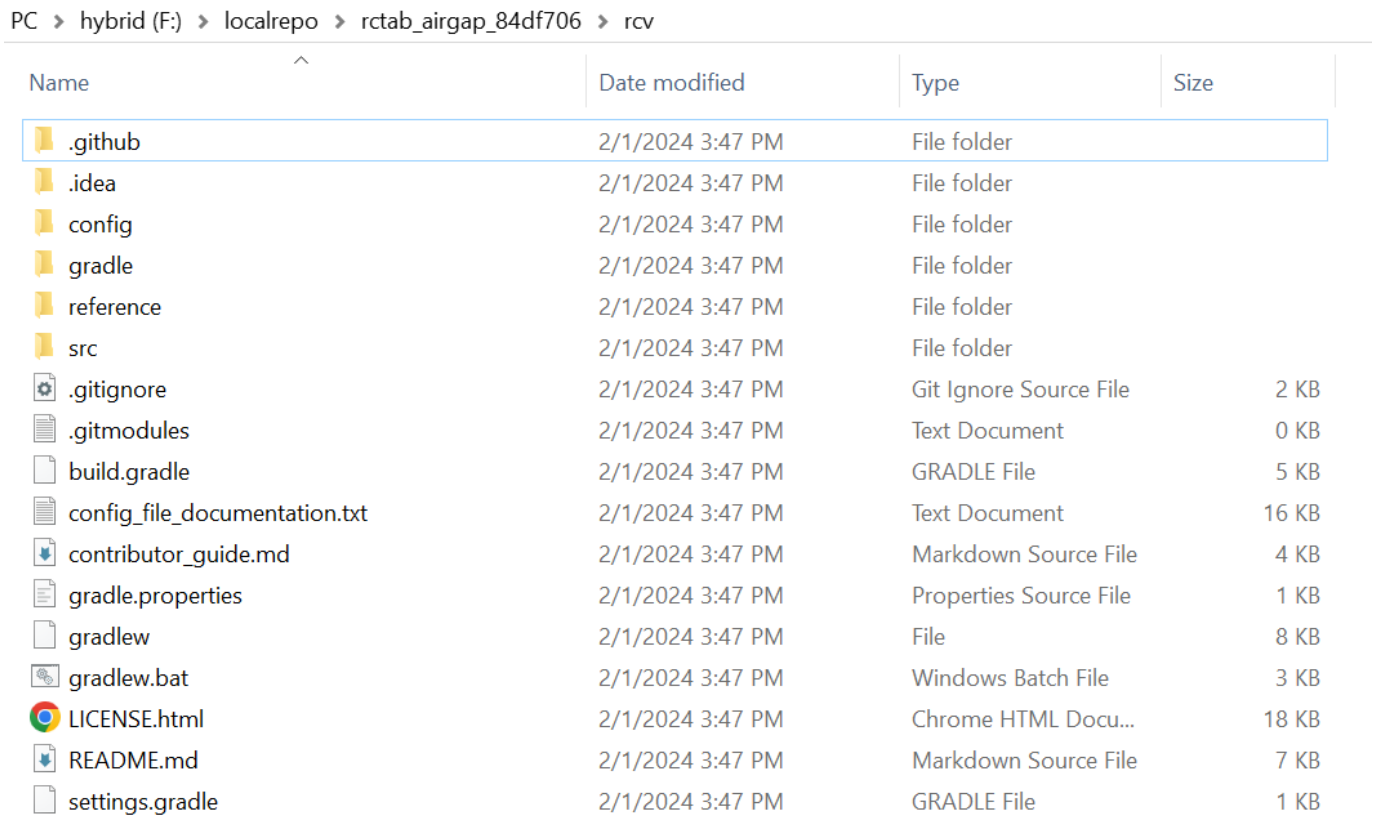
1. If Git is not already present on the machine, install it. The Windows version of Git is available here: https://github.com/git-for-windows/git/releases/tag/v2.42.0.windows.2

a. `X64` git installer is `Git-2.42.0.2-64-bit.exe` with a `SHA256` of `bd9b41641a258fd16d99beecec66132160331d685dfb4c714cea2bcc78d63bdb`

b. Run through the installer with all default settings

2. Open Command Prompt
3. Enable the autocrlf setting by running this command in the command prompt: `git config --global core.autocrlf true`
4. Clone the repository with this command: `git clone --branch https://github.com/BrightSpots/rcv.git`

   For the rest of this example we'll refer to the folder that your source code is in as `.\` In the example below, that corresponds to `f:\localrepo\rctab_airgap_84df706\rcv\`



JAVA

Java JDK

Building the application will also require the proper version of the Java development kit to be located on the machine.

- Navigate to https://jdk.java.net/archive/.
- Download the .zip for the Windows x64 version of OpenJDK 17.0.2 (build 17.0.2+8).
- The SHA256 for the build 17.0.2+8 Windows x64 zip is `b2208206bda47f2e0c971a39e057a5ec32c40b503d71e486790cb728d926b615`
- The .zip file when downloaded is named `openjdf-17.0.2_windows-x64_bin.zip`
- Extract the `openjdk-17.0.2_windows-x64_bin.zip`. When unzipped, the folder it creates will be named `jdk-17.0.2`. Note the location of this folder for the next step

Linux/x64	**64-bit**	tar.gz (sha256) 179M
	Source	Tags are jdk-18.0.1.1+2, jdk-18.0.1.1-ga

18 GA (build 18+36)		
Windows	**64-bit**	zip (sha256) 178M
Mac/AArch64	**64-bit**	tar.gz (sha256) 174M
Mac/x64	**64-bit**	tar.gz (sha256) 176M
Linux/AArch64	**64-bit**	tar.gz (sha256) 177M
Linux/x64	**64-bit**	tar.gz (sha256) 179M
	Source	Tags are jdk-18+36, jdk-18-ga

17.0.2 (build 17.0.2+8)		
Windows	**64-bit**	zip (sha256) 178M
Mac/AArch64	**64-bit**	tar.gz (sha256) 174M
Mac/x64	**64-bit**	tar.gz (sha256) 176M
Linux/AArch64	**64-bit**	tar.gz (sha256) 178M
Linux/x64	**64-bit**	tar.gz (sha256) 179M
	Source	Tags are jdk-17.0.2+8, jdk-17.0.2-ga

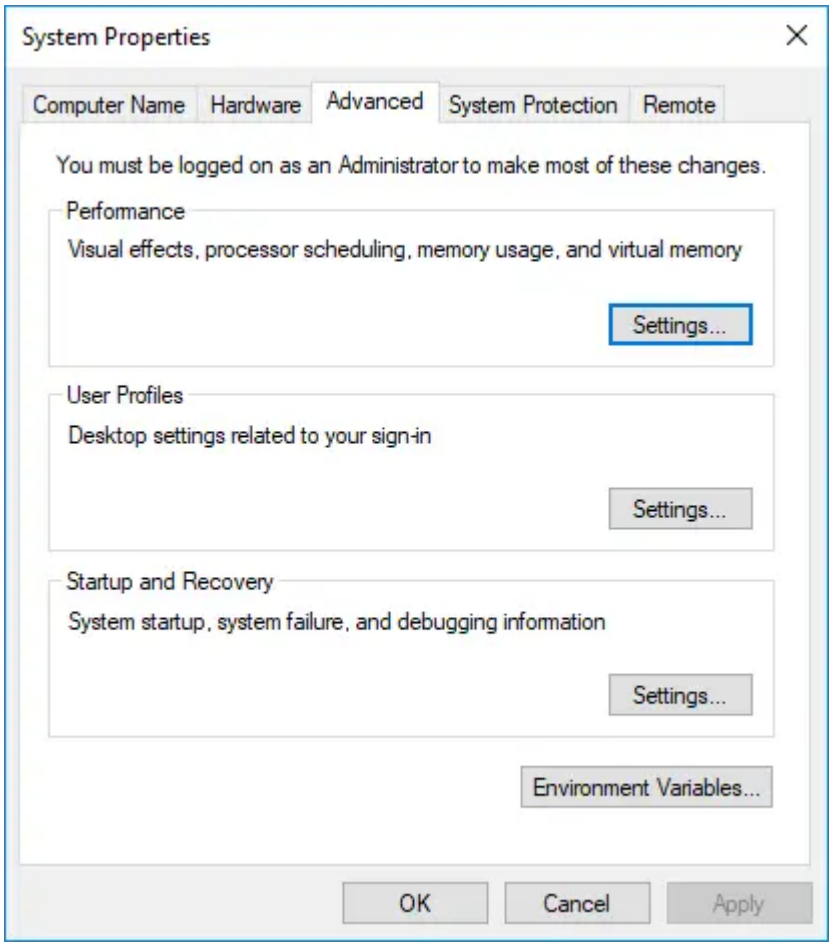
l7.0.1 (build 17.0.1+12)		
Windows	**64-bit**	zip (sha256) 178M
Mac/AArch64	**64-bit**	tar.gz (sha256) 174M
Mac/x64	**64-bit**	tar.gz (sha256) 176M
Linux/AArch64	**64-bit**	tar.gz (sha256) 177M

Note that you must use this exact version of the JDK. If you obtain the JDK from a different location, even if the version and build numbers are the same, it may have subtle differences that will prevent your build from matching ours via the process we've developed. Specifically, we've determined that the Oracle-licensed build of OpenJDK is not identical to the London Jamocha Community CIC-licensed build.

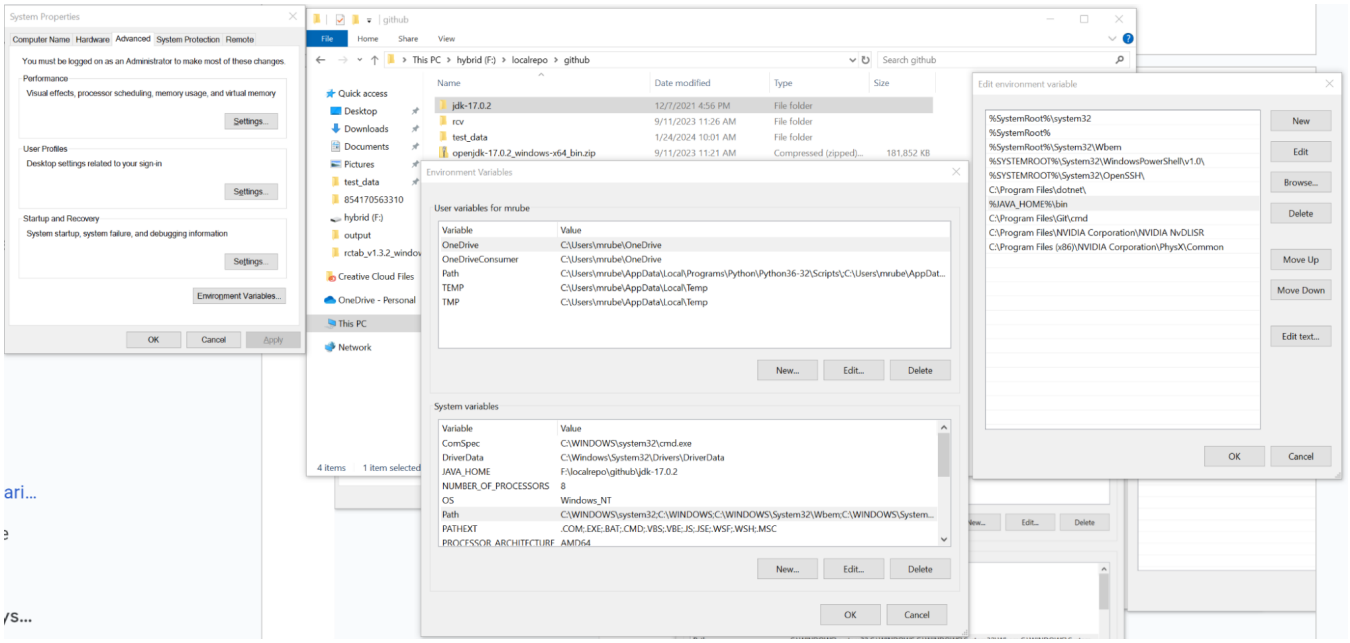
Set JAVA_HOME variable

Use the following instructions to set your JAVA_HOME variable, a requirement for building the source code

- Go to Start Menu and search for “Advanced System Settings”
- Click ‘View Advanced System Settings’
- Go to the Advanced Tab and click ‘Environment Variables’ button



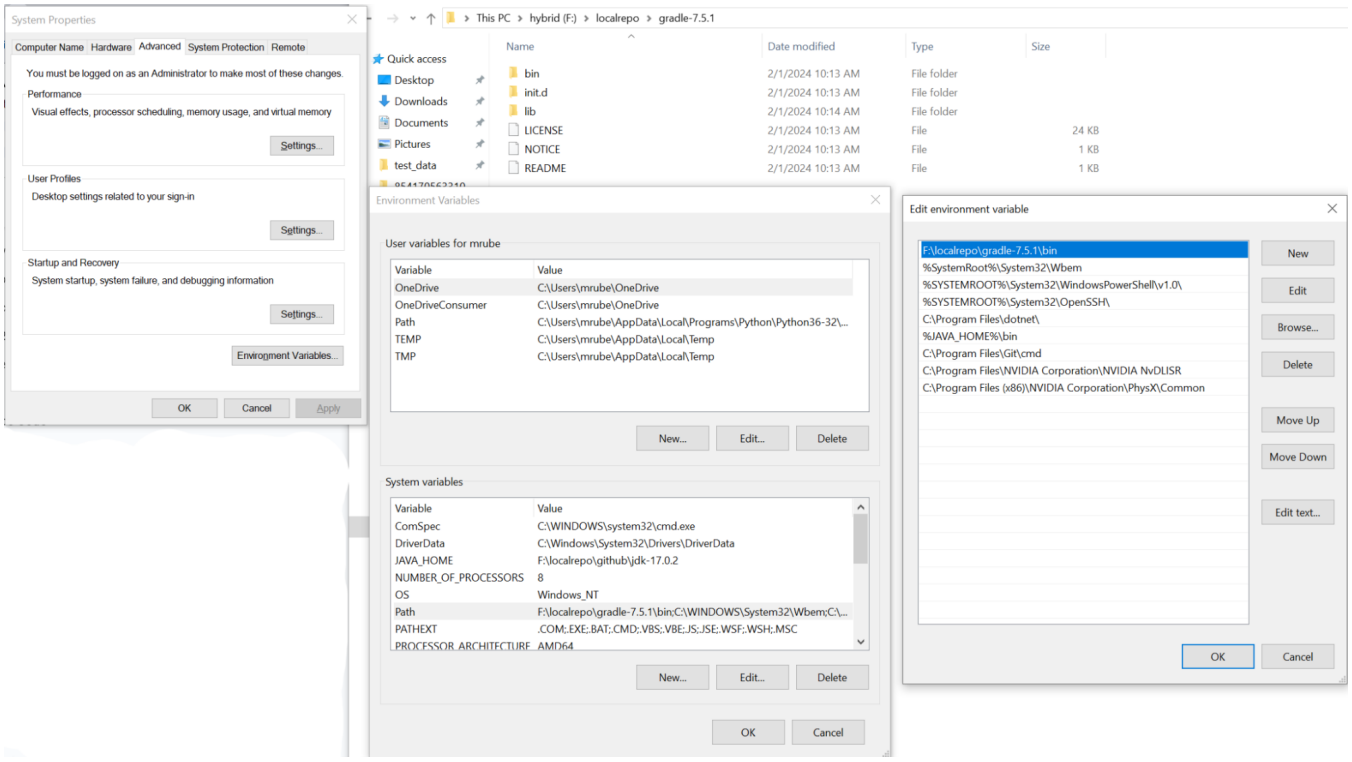
- In the Environment Variables window that pops up, in the bottom **System variables** section click ‘New’ and add `JAVA_HOME` as the Variable name and click ‘Browse Directory’ to browse to your **unzipped** java folder `jdk-17.0.2` and set it’s path as the Variable value



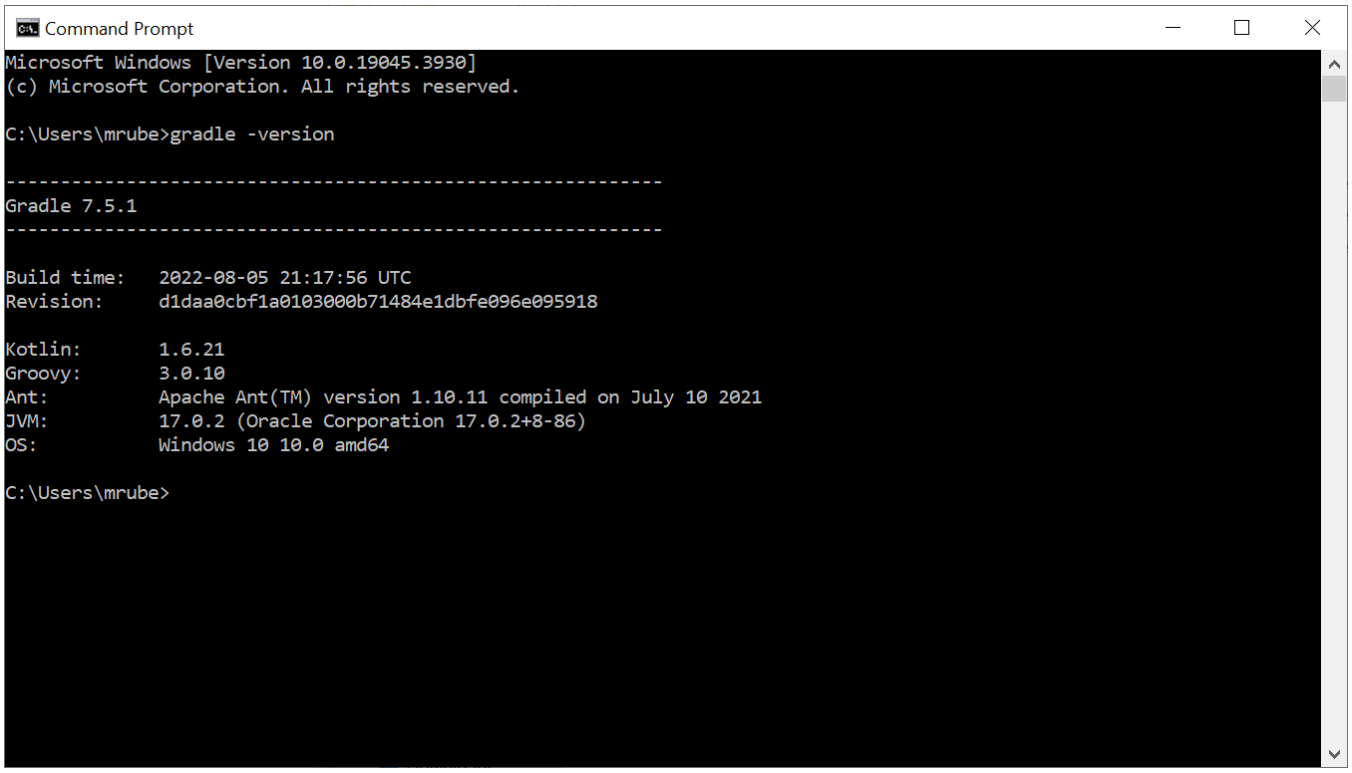
- Go back to the Environment Variables window, find the `Path` entry in the bottom System Variables section and click edit
- Click ‘New’ and add `%JAVA_HOME%\bin`
- Click ‘Ok’
- Click ‘Ok’ in Environment Variables window
- Click ‘Ok’ in System Properties window
- Close any Command Prompt Windows you have open
- Re-open Command Prompt and confirm that Java 17.0.2 is properly installed by running this command in the cmd prompt `java –version` you should see an entry for `openjdk 17.0.2 2022-01-18`

GRADLE

- Download Gradle 7.5.1 from the Gradle release page
- Download the ‘binary-only’ option. This should download `gradle-7.5.1-bin.zip` with a hash of `f6b8596b10cce501591e92f229816aa4046424f3b24d771751b06779d58c8ec4`
- Extract the zip. Add the `bin` folder of the extracted zip to your System Path like you did for Java



- Open Command Prompt and type `gradle -version` to confirm

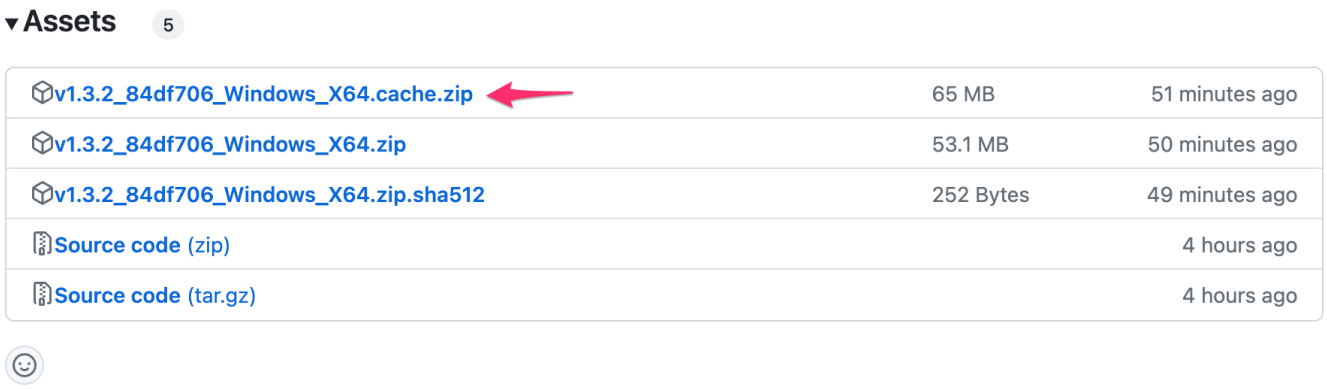


- Type `gradle -stop` in Command Prompt.

THIRD PARTY DEPENDENCIES

The final step before building requires you to have the third party dependencies that RCTab relies on.

- Download the .zip of all necessary dependencies. The zip of all necessary dependencies for the v1.3.2 California release is called `v1.3.2_84df706_Windows_X64.cache.zip` and can be downloaded from the release page under 'Assets'



- Extracting the .zip should create a `modules-2` folder. Move that folder to `~\.gradle\caches\` If that folder doesn't exist you should create it. `~\` in this step represents the logged-in Windows User folder. For example, if my Windows Username is `RCTabUser` then at the end of this step I would have `C:\users\RCTabUser\.gradle\caches\modules-2`
- You can check which folder to substitute `~\` for by typing `echo %userprofile%` into Command Prompt
- If required, use jurisdiction rules to verify dependencies with `~\.gradle\caches\checksums.csv`. This file contains a line for each necessary dependency that has
- Filename
- SHA-1 Checksum
- SHA-256 Checksum
- Maven Dependency URL
- Direct URL to SHA-1
- Direct URL to SHA-256

Building From Source

Next, navigate into the source code directory in Command Prompt. We will run a command that will build the RCTab source code along with all of the third party dependencies (which are listed in **Section 02 - Software Design and Specifications**). This command will build RCTab from source using Java 17.0.2 and Gradle 7.5.1

In Command Prompt, type the Gradle build command for Windows:
`gradle jlinkZip`

The output of the Gradle build command is a single ZIP file called `rcv.zip` that will be located in the `.\build\` subdirectory. To complete the build process, rename the ZIP file to reflect the platform and version of the application.

On Windows, building version 1.3.2, for example:
`rename build\rcv.zip rctab_v1.3.2_windows.zip`

To generate a hash signature for the completed build, run this command on Windows:
`C:\Windows\System32\certutil.exe -hashfile build\rctab_v1.3.2_windows.zip sha512`

Example output on Windows:

```
SHA512 hash of build\rctab_v1.3.2_windows.zip:
f1b05aa5cdff8c01ee220579d5b5310911a75d0e58bc5568113b893845a31c47abab8a0875bda78f29a89e8218d22af00ea770c83ff39eb5b68b41ddfb8254ee
CertUtil: -hashfile command completed successfully.
```

Note, however, that two independent runs of the `jlinkZip` task will result in ZIP files with different hashes! Although the file contents in the two ZIP files will be identical, the timestamps of those files will not be. Those timestamps are included in the hash and thus, the hashes will not match. As a result, generating a hash for the ZIP file itself is only suitable for verifying that **this build has not been modified.**

To confirm that the file *contents* of one build match those of another, for example that the local build matches the official release on the BrightSpots github, see 'Comparing Two Builds' below.

2.19.3 Comparing two builds

As noted above, two independent builds of the same source code will have different hashes due to the presence of file timestamps within the ZIP file. It is, however, possible to demonstrate that two builds are equivalent by stripping the timestamps.

The process for generating a single timestamp-independent hash signature for a build ZIP requires a few steps: not only does the ZIP file store timestamps for the files it contains, but one of the files within the ZIP file, `lib/modules`, is, in turn, a collection of time stamped files. Fully decomposing the build to its individual constituent files thus requires using the `jimage` utility (part of the Java 17 development kit) to extract the files contained within `lib/modules`.

- First, unzip the ZIP you built locally (by following the precise instructions in the "Full build" section above). For the rest of this example we'll assume that you extracted the zip files to `c:\rctab_v1.3.2_windows\`
- **Make sure not to execute any other commands that create, modify, or delete files within this directory (like opening the application). Any modifications to these files will change the result of the hashing process.**
- Create a `hash.bat` file in the `c:\rctab_v1.3.2_windows\` folder with the following command `echo.>hash.bat`
- Open hash.bat in notepad and paste in the text below these bullets (starting with ::hash.bat)
- Run the batch file in the cmd prompt by typing `hash.bat` then enter. **It can take 30-45 minutes to complete as it is computing a hash for every single file**

```
:: hash.bat
@echo off

:: NOTE: This script must be placed one level up from the rcv directory

echo Initiating batch hash procedure
set startTime=%date% %time%
echo %startTime%

setlocal EnableExtensions EnableDelayedExpansion

set "HASHFILE=all_hashes.txt"
set "TEMPHASHFILE=all_hashes_temp.txt"
set "EXTRACTIONDIR=.\rcv\modules_extracted"
set "MODULESFILE=.\rcv\lib\modules"

if exist %HASHFILE% (
    echo Deleting existing hash file, %HASHFILE% ...
    del %HASHFILE%
)

if exist %EXTRACTIONDIR% (
    echo Deleting existing extracted modules directory, %EXTRACTIONDIR% ...
    rmdir /s /q %EXTRACTIONDIR%
)

echo Extracting contents of modules file...
mkdir %EXTRACTIONDIR%
cd %EXTRACTIONDIR%
jimage extract ..\..\%MODULESFILE%

echo Temporarily relocating modules file...
cd ..\..
copy %MODULESFILE% .
del %MODULESFILE%

:: Calculate the hash for every file here and in all subdirectories, appending to the file (format "(filename) = (hash)")
echo Calculating hashes...
for /r .\rcv %%f in (*) do (
    <NUL set /p ="%%f = " >> %HASHFILE%
    C:\Windows\System32\certutil.exe -hashfile "%%f" SHA512 | findstr /v ":" >> %HASHFILE%
    echo hashing %%f
)

echo Restoring modules file...
move .\modules %MODULESFILE%

:: Replace the absolute paths to each file with relative paths (e.g. C:\temp\rcv => .\rcv)
```

```
echo Replacing absolute paths with relative paths in hash file...
set "SEARCHTEXT=%cd%"
set "REPLACETEXT=."
for /f "delims=" %%A in ('type "%HASHFILE%"') do (
    set "string=%%A"
    set "modified=!string:%SEARCHTEXT%=%REPLACETEXT%!"
    echo !modified!>>"%TEMPHASHFILE%"
)
del "%HASHFILE%"
rename "%TEMPHASHFILE%" "%HASHFILE%"

echo Sorting the hash file...
sort "%HASHFILE%" > "%TEMPHASHFILE%"
del "%HASHFILE%"
rename "%TEMPHASHFILE%" "%HASHFILE%"

echo Calculating the hash of the entire sorted hash file...
C:\Windows\System32\certutil.exe -hashfile %HASHFILE% SHA512

endlocal
```

Example output (Windows):

```
SHA512 hash of all_hashes.txt:
dfc48c8e8606316144ebbcb2e7c5dacfdc709834460b443a5a88a9e09ea7273b59c1a4d3605fea3654e57aa11143461b2c8f035bd380af388ecc0fb513b98c7e
CertUtil: -hashfile command completed successfully.
```

2.20 Section 15 - System Change Notes

2.20.1 9.12 System Change Notes

Manufacturers submitting modifications for a system that has been tested previously and received national certification shall submit system change notes. These will be used by the S-ATA to assist in developing and executing the test plan for the modified system. The system change notes shall include the following information:

1. Summary description of the nature and scope of the changes, and reasons for each change
2. A listing of the specific changes made, citing the specific system configuration items changed and providing detailed references to the documentation sections changed
3. The specific sections of the documentation that are changed (or completely revised documents, if more suitable to address a large number of changes)
4. Documentation of the test plan and procedures executed by the manufacturer for testing the individual changes and the system as a whole, and records of test results

2.20.2 RCTab Version 1.3.2

RCTab version 1.3.1 was submitted with Hart InterCivic's Verity Voting 3.2 to SLI for initial testing to California Voting System Standards. After an initial round of testing we received SLI's report. Some of the report results identified recommendations for RCTab mostly related to access control and programmatically verifying RCTab input and output. Ranked Choice Voting Resource Center (RCVRC) submitted a new version, v1.3.2, to address each of the report results. v1.3.2 updates apply to Windows only. **In 2.0**, Hart Cryptographic Signature verification is not enabled by default as it would only work in the state of California.

New Features

- Two Windows OS accounts. One Administrator level Windows account for installation and initial configuration. A separate 'RCTab' Windows standard account for running the tabulation with only necessary permissions and **will not have access to make administrative changes to the machine** that could be security issues
- Read-Only RCTab output files, each with a corresponding read-only `.hash` file that contains the hash of file contents to ensure they haven't been edited
- Automatic, programmatic verification of the cryptographic signature of Hart input CVRs. This verification step ensures both the integrity (CVR contents have not been edited) and provenance (CVRs came from the Hart voting system) of the Hart CVRs. RCTab will throw a halting error and tabulation will not begin if cryptographic validation of Hart's CVR signature is not successful.

Hardware Setup Improvements

- Explicit TDP directions for a secure, hardened OS install
- Password secured BIOS

TDP Updates

- **Section 02 - Software Design and Specifications**
- Includes `.hash` file descriptions in **Reporting Results** header, **RCTab Logging Stream 2: Contest-Specific "Audit" Logging** header, `ResultsWriter` Java class
- Removes paragraph in **RCTab Logging - Stream 2: Contest-Specific "Audit" Logging** header about config validation missing from audit log. That was fixed in 1.3.1
- 9.5.3.c.3 updated with link to **Section 14 - Tabulator Trusted Build Instructions**
- 3rd Party Modules section updated with new version numbers and added `BouncyCastle` FIPS API module
- Include new 1.3.2 Java classes: `AuditableFile`, `SecurityConfig`, `SecuritySignatureValidation`, `SecurityXmlParsers`, `SecurityTests`

- **Section 07 - System Security Specification Requirements**
- Add RCTab Windows standard user account, read-only outputs, `.hash` files for RCTab output and Hart CVR signature verification throughout CVSS security & access control requirements
- 9.6.1.a, 9.6.1.b, 9.6.1.c, 9.6.1.d, 9.6.1.e, 9.6.1.1, 9.6.1.1.a, 9.6.2
- 9.6.1.b updated to include Windows OS Account and BIOS Password access control mechanisms
- 9.6.1.2 updated to include full description of two tier Windows OS accounts for access control
- 9.6.1.1.e updated to include explicit overview of Operating System protection abilities
- 9.6.3.f describes in detail the signature verification interaction with Hart Verity
- 9.6.7 Cryptography header updated with explicit cryptographic signature verification details
- **Section 07I - Design and Interface Specification**
- Includes all security enhancements for 1.3.2 and describes the threat that they protect against as well as the threats that still exist
- Added explicit system security objectives and known vulnerabilities
- **Section 07J - Security Architecture**
- Include 1.3.2 updates in Access Control and Integrity sections
- **Section 07L - Security Threat Analysis**
- Include all 1.3.2 updates and describe the threats that they protect against
- **Section 07M - Security Testing and Vulnerability Analysis**
- Include all 1.3.2 updates to describe how initial testing report results have all been addressed
- **Section 09 - System Maintenance Manual**
- Includes read-only output and `.hash` file verification in 9.9.1.f
- **Section 11 - L&A Testing**
- Update hash verification of output with `.hash` files
- **Section 12 - Configuration Management Plan**
- Update RCTab Workstation model, hash value of v1.3.2, required software
- **Section 13 - Quality Assurance Plan**
- Adds security tests for v1.3.2
- **Section 14 - Tabulator Trusted Build Instructions**
- Clarify Java install check
- Add step to confirm gradle install
- Clarify `jlinkZip` should be done in the cmd prompt
- v5 - Add steps for offline build
- Third party dependencies cache
- Gradle
- v5 - Fix `--branch` instead of `-branch` in git clone command
- v5 - Clarify git instructions to use specific version
- v5 - Clarify instructions for `JAVA_HOME` variable, `hash.bat`
- Added explicit instructions for setting `JAVA_HOME` variable
- Updated Comparing Two Builds section to explain more clearly how we get a single hash, remove ‘Individual Files’ section
- **Section 15 - System Change Notes**
- Include code updates, TDP updates

- **Section 16 - System Hardening Procedures - Windows OS**
- Clarify and clean up Windows OS install steps
- Remove UPS step
- Included BIOS password, Windows OS Installation
- Make clear the distinction between online prep and offline updates
- Explicit steps for enabling Bitlocker device encryption
- Edit 'Operating System Hardening' header to disable screensaver and correctly disable Remote Desktop
- Remove additional optional antivirus, Windows Update
- Replace Excel with LibreOffice
- Remove manual driver installation
- **Section 17 - System Test and Verification Specification**
- Adds security tests for v1.3.2
- **Section 18 - User Guide**
- Include automatic signature verification of Hart signed CVRs, `.hash` output files and read-only output.
- Output Directory instructions detail explicit steps to ensure read-only permissions are set
- Include directions to configure Verity Count to sign CVR exports
- Remove manual hash steps in 'Generating Results Files' header
- Programmatic CVR signature verification in Hart CVR Files Options header and Prepping CVRs For Use header
- Instructions for Read-Only output in Generating Results Files - Output Tab header
- Running A Tabulation header output files updated to include `.hash` files
- Remove CVR `.csv` from list of output files as it doesn't apply to Hart
- **Section 22 - Installation Instructions for Windows OS**
- Include instructions for creating 'RCTab' Windows Standard User
- Explicit instructions for RCTab Windows Standard user vs. Windows Administrator user
- Include directions for enforcing read-only permissions on output folder
- Include directions for creating desktop shortcut
- **Section 23 - Trusted Build & Output Hash Verification - Windows OS**
- Use `.hash` files for output summary file verification. Use different example path that doesn't use user folders
- **Section 24 - Tabulator Command Line Instructions**
- Use different example path outside of user paths
- **Section 26 - RCTab CVR Files**
- Include .sig.xml file description for required Hart CVR archive cryptographic signature
- **Section 28 - Post-Election Audit & Clearing RCTab from System**
- Include comparing the text of output file `.hash` files to the hash text in the audit log as another auditable check
- **Section 29 - RCTab Operator Log Messages**
- Added additional v1.3.2 Severe messages

2.20.3 RCTab Version v.1.3.1

Bug fixes:

- Fixed XML parsing failing when running built version (#625)

Backend updates:

- Releases for all platforms are now automatically built by GitHub when published (#282)

2.20.4 RCTab Version v.1.3.0

New features:

- Added support for multi-file Dominion format (#569)
- Allows batch elimination and "continue until two candidates remain" to be enabled in multi-pass IRV mode (#611)
- Allows users to specify multiple CVR files at once in the GUI (#617)
- Adds validation highlighting to the GUI when clicking the "add" buttons for candidates and CVRs (#618)
- Changed audit logs to include validation outcome (#616)

Bug fixes:

- Fixed Hare Quota (#562)
- Fixed build for M1 Macbooks (#586)
- Fixed crashes when % was in file paths (#601)
- Fixed inaccurate `overvoteRule` error message (#609)
- Fixed bug in logic for exhaustIfMultipleContinuing overvote rule (#610)
- Fixed bug where `treatBlankAsUndeclaredWriteIn` validation failure for certain providers wouldn't actually fail validation (#618)

Other improvements

- Rebranded "Universal RCV Tabulator" as "RCTab" (#603)
- Updated license from AGPL to MPL 2.0 (#604)
- Exits gracefully if all declared candidates fall beneath minimum vote threshold (#608)
- Moved validation of provided overvote delimiter and overvote label into `performBasicCvrSourceValidation` so user is alerted when clicking the "add" button in the CVR tab (#618)
- Updated documentation and help text (#547, #614, #617)

Backend updates:

- Enabled CI, which runs tests, Checkstyle, and Spotbugs (#576)
- Addressed all outstanding Checkstyle and Spotbugs warnings (#587)
- Internal clean-up to conform to VVSG coding requirements (#600, #602, #606)
- Changed `ContestConfig.validate()` and associated methods to return a set of validation errors instead of an `isValid` boolean (#618)
- `checkstyle-suppressions.xml` location is now handled in `build.gradle` to avoid needing to manually modify `google_checks.xml` in the future (#544)

- Updated dependencies to latest versions:
- JDK 17.0.2
- JavaFX 18
- Gradle 7.5.1
- Checkstyle `google_checks.xml` 10.3.2
- Checkstyle plugin 10.3.2
- spotbugs 4.7.1
- `spotbugs-gradle-plugin` 5.0.9
- `org.openjfx.javafxplugin` 0.0.11
- `org.beryx.jlink` 2.25.0
- `com.fasterxml.jackson.core:jackson-*` 2.13.3
- `org.junit.jupiter.junit-jupiter-*` 5.9.0
- `org.apache.commons:commons-csv` 1.9.0
- `org.apache.poi:poi-ooxml` 5.2.2

2.20.5 Universal RCV Tabulator (RCTab) Version v.1.2.0

Certification:

Based on the testing performed by Pro V&V, a certified Voting System Testing Laboratory, and the results obtained, the modified RCV-Tabulator solution identified in this test report meets the requirements set forth by the VVSG 1.0.

New features:

- Added support for new manufacturer formats:
- Clear Ballot (#400)
- Dominion (#119, #438, #533)
- Hart (#401, #457, #460)
- Unisyn (#402)
- Redesigned GUI to be more user-friendly (#461, #123, #128, #152) (see "GUI redesign" section below for more details)
- Added support for "Overvote Delimiter" field (#482)
- "Minimum Vote Threshold" field is now optional (#483)
- "Undeclared Write-In Label", "Overvote Label", "Undervote Label", and "Treat Blank as Undeclared Write-in" fields are now defined on a per-CVR level (#508)

GUI redesign:

- Added hint panels for each tab (#499)
- Added help menu option for the config documentation (#497, #528)
- Changed "Continue until Two Candidates Remain" to a boolean config setting (#481)
- Combo text box / check box input for "How Many Consecutive Skipped Ranks Are Allowed" and "Maximum Number of Candidates That Can Be Ranked" (#498)
- Disabled "Decimal Places for Vote Arithmetic" in non-multi-seat modes (#500)
- Redesigned `nonIntegerWinningThreshold` and `hareQuota` as a three-way radio button and added new validation rules (#501)
- Disabled editing existing candidates and CVR sources after adding to prevent confusing UX (#502)
- Added support for multiple contests via implementation of "Contest ID" field (#456, #472, #478)

ADDITIONAL GUI CHANGES:

- Split Output tab into new Contest Info and Output tabs
- Redesigned GUI CVR Files tab, adding Clear button, and changing Add button, so it only enter multiple sources that share fields clears the file path to make it easier to manually
- Improved visual presentation of Candidate tab; added Clear button and adds `checkBoxCandidateExcluded` when adding a candidate
- Reorganized presentation of rules in "Winning Rules" and "Voter Error Rules" tabs
- Winner Election Mode and Tiebreak Mode now start undefined with all relevant fields disabled; choosing specific modes enables applicable fields
- Changed Winner Election Modes and Tiebreak Modes to be more user-friendly, including necessary migration logic to update older config files
- Expanded footprint of GUI window to 1200x1000
- Implemented bordered boxes
- Converted `overvoteRule` from a ChoiceBox to an array of `RadioButtons`; changed `overvoteRule` string display in config files and adds migration logic
- Disabled `decimalPlacesForVoteArithmetic` and `nonIntegerWinningThreshold` except when `winnerElectionMode` is "Multi-winner allow only one winner per round" or "Multi winner allow multiple winners per round" (fixes #500)
- Added suggested values for `overvoteLabel`, `undervoteLabel`, and ES&S column and row indices
- Replaced `checkBoxNonIntegerWinningThreshold` and `checkBoxHareQuota` with a radio button array, and added new validation rules for those settings (fixing #501)
- GUI now disables `numberOfWinners` field and sets it to `1` only when `winnerElectionMode` is "Single-winner majority determines winner"
- GUI now disables `numberOfWinners` field and sets it to `0` when `winnerElectionMode` is "Bottoms-up using percentage threshold"
- Fixed bugs in validation error messages when `numberOfWinners` is `0`

Bug fixes:

- Fixes CDF JSON reading and writing (#505)
- Fixed being unable to tabulate multiple CDF sources (#536)
- Fixed user and computer name logging (#521)
- Fixed config referencing nonexistent CDF source leading to uncaught exception (#347)
- Fixed incorrect overvote label in CDF leading to NPE (#453)
- Fixed config file with bad provider value failing with NPE (#531)

Other improvements:

- Removed "Convert Dominion to Generic Format..." functionality since direct Dominion tabulation is now possible (#476)
- Registers explicit overvote as valid candidate / contest selection in CDF output when needed (#451)
- Handles bad path to CDF CVR source gracefully (#452)
- Handles empty rows at end of CVR (#455)
- Made sure all providers work with the CLI (#471)
- Raises error if we encounter an unrecognized candidate while loading Dominion CVRs during direct tabulation (#473)
- Reports error if config specifies any column indexes for a CDF source (#276)

Backend updates:

- Created separate `MigrationHelper` class (#507)
- Addressed warnings during Gradle build (#280)
- Upgrading to a more recent version of Gradle no longer causes test failures (#283)

- Addressed all relevant Checkstyle warnings and disabled all invalid ones (#490, #489)
- Enum parameters now use camel case for backend and user-friendly strings for frontend (#494)
- Fixed noinspection unchecked for excluded `CheckBox` in `GuiConfigController` (#304)
- Now use `UnrecognizedCandidatesException` in `ClearBallotCvrReader` and `HartCvrReader` (#491)
- Updated dependencies to latest version:
- JDK 14.0.1
- JavaFX 14.0.1
- Gradle 6.5.1
- Checkstyle `google_checks.xml` 8.36.2
- Checkstyle plugin 8.36.2
- `org.openjfx.javafxplugin` 0.0.9
- `org.beryx.jlink 2.20.0`
- `com.fasterxml.jackson.core:jackson-*` 2.11.1
- `org.junit.jupiter.junit-jupiter-*` 5.6.2

2.20.6 Universal RCV Tabulator (RCTab) Version v.1.1.0

Certification:

Based on the testing performed by Pro V&V, a certified Voting System Testing Laboratory, and the results obtained, the Universal RCV Tabulator v1.1.0 solution is believed to meet the applicable requirements set forth by the EAC-approved VVSG 1.0.

New features:

- Added support for converting Dominion JSON CVRs to generic `.csv` format (including precinct portions) (#404, #406, #407, #408, #415, #439)
- Added `multiSeatBottomsUpPercentageThreshold` option (#403)
- Added CLI option to convert Dominion CVR to generic `.csv` (#408)
- New GUI menu and conversion options (can now convert to CDF and convert Dominion to generic via the GUI) (#408, #421)
- Added Dominion Alaska CVR to `sample_input` folder

Bug fixes:

- Batch elimination now works properly with `singleSeatContinueUntilTwoCandidatesRemain` (#396)
- In a multi-seat contest, if someone wins in the first round, we now automatically eliminate undeclared write-ins before we eliminate any other candidates; previously, we treated UWIs like a normal candidate, which meant we potentially eliminated other candidates with lower tallies first (#397)
- If UWI exceeds the winning threshold in the initial count, we no longer mistakenly elect this candidate (#398)

Backend updates:

- Updated dependencies to latest version:
- JDK
- JavaFX
- Checkstyle `google_checks.xml`
- Checkstyle plugin
- `org.openjfx.javafxplugin`
- `org.beryx.jlink`
- `org.apache.commons:commons-csv`
- `org.apache.poi:poi-ooxml`
- `com.fasterxml.jackson.core:jackson-*`
- Added special code to test configs to obviate the need to update the version with each increment (#426)
- Updated tests and improved test coverage
- Copyright update (#414)
- Code cleanup

2.20.7 Universal RCV Tabulator (RCTab) Version v.1.0.1

Based on the testing performed by Pro V&V, a certified Voting System Testing Laboratory, and the results obtained, the Universal RCV Tabulator solution meets the requirements set forth by the EAC-approved VVSG 1.0 to be used with the ES&S EVS `5.0.0.0` through `6.0.4.0` software.

Changelog:

- Added Checkstyle plugin to Gradle and set it up for Google format
- Minor refactoring to address Checkstyle issues

2.21 Section 16 - System Hardening Procedures - Windows OS

The hardware used to house RCTab 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 (i.e. Windows) 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 OS, disabling automatic windows login, and proper configuration of the system among other things. No unauthorized software should be installed on the computer unless authorized by your jurisdiction. All RCTab systems must include only the following software:

- Windows 10 Pro (or newer)
- RCTab v1.3.2
- LibreOffice 7.5.7 - For viewing formatted cvr output
- XML Viewer - For viewing CVR information
- Users must also retain access to:
- Command Prompt
- Notepad
- UPS

See below for procedures for securing verified versions of COTS software listed above.

Note: 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 employees) should have physical access to the standalone computer.

The computer should be in a secure location.

2.21.1 Required Items for System Setup & Hardening

Jurisdictions should have the following artifacts for System Setup & Hardening

- HP Z4 G4 workstation
- USB 1 - contains HpSetup.txt file for Bios Password setup
- USB 2 - contains bootable Windows 10 image

Note: Jurisdiction should follow secure password practices and provide access to the least amount of users necessary for the BIOS password and Windows OS Admin password created in the following steps. These passwords shall be used for installation and initial configuration. Users running the tabulation shall not have access to these passwords.

2.21.2 Bios Password

The following steps will set a BIOS password and relevant boot security settings to harden the computer.

1. Power on the workstation.
2. Press F10 repeatedly until the BIOS menu appears.
3. Create a Workstation BIOS password for the workstation.
4. Navigating to “Security” → “Create a BIOS Administrator Password”: Enter a Workstation BIOS password (at least 8 characters. Uppercase, Lowercase, 0-9, and symbol characters are all acceptable)

a. Enter the password and press Enter.

b. Enter the password again (to verify), and press Enter.

c. **Please Note: There is no recovery to reset and re-enter the BIOS if the BIOS password is forgotten or lost.**

5. Insert a USB Stick with the Replicated Setup `HpSetup.txt` file.
6. Go to “Main” → “Replicated Setup.”
7. Click “Restore current settings from USB device.”
8. Click “Browse for Configuration File” and navigate to the HpSetup.txt on the USB 1 stick.
9. A prompt will appear that says “Number of settings successfully restored (239). Number of settings that were invalid (0).” Hit “OK.”
10. Press F10 to save changes and exit.
11. The system will reboot multiple times. Confirm the settings change when prompted during the reboot processes.

2.21.3 Windows OS Install

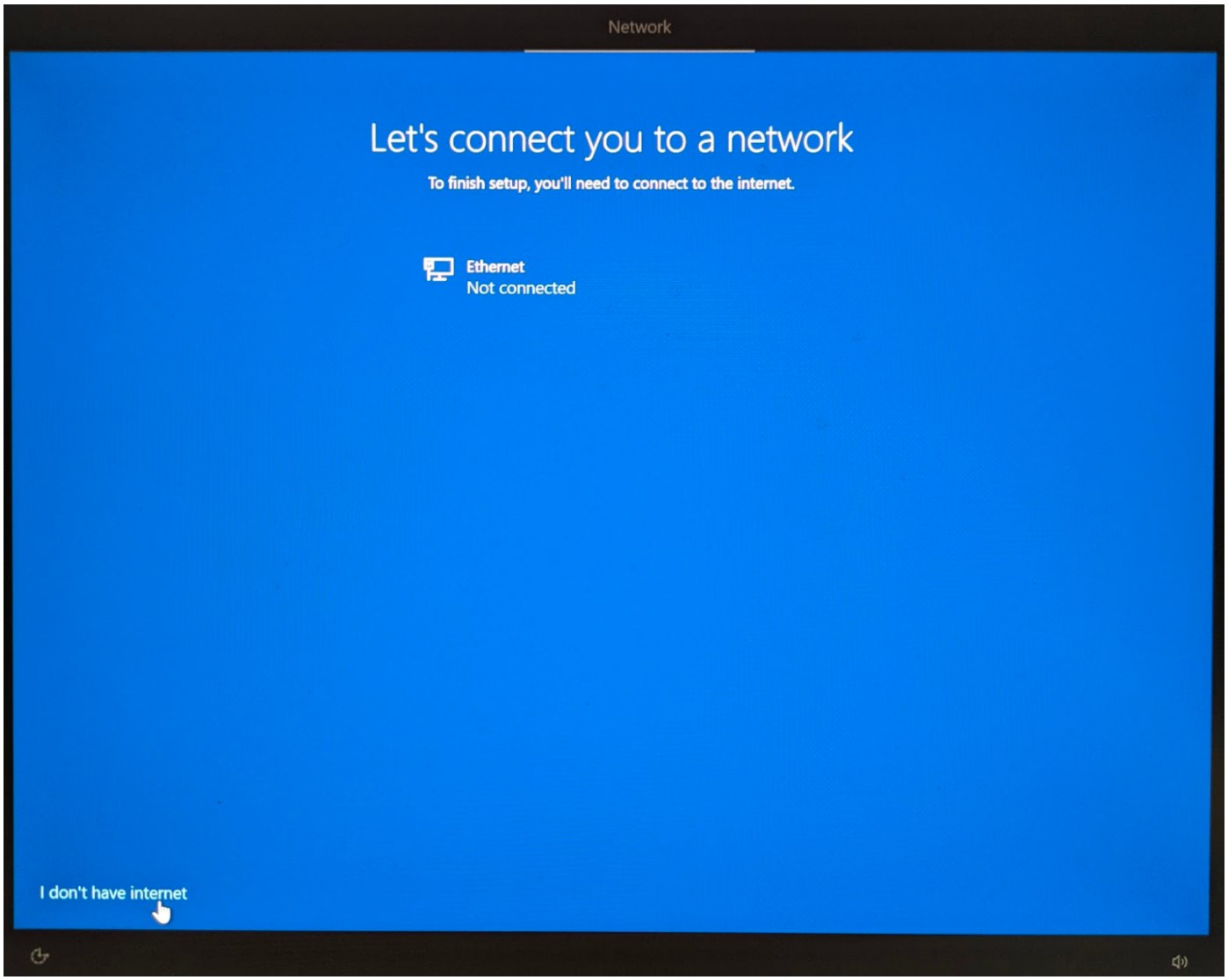
First, enable USB boot.

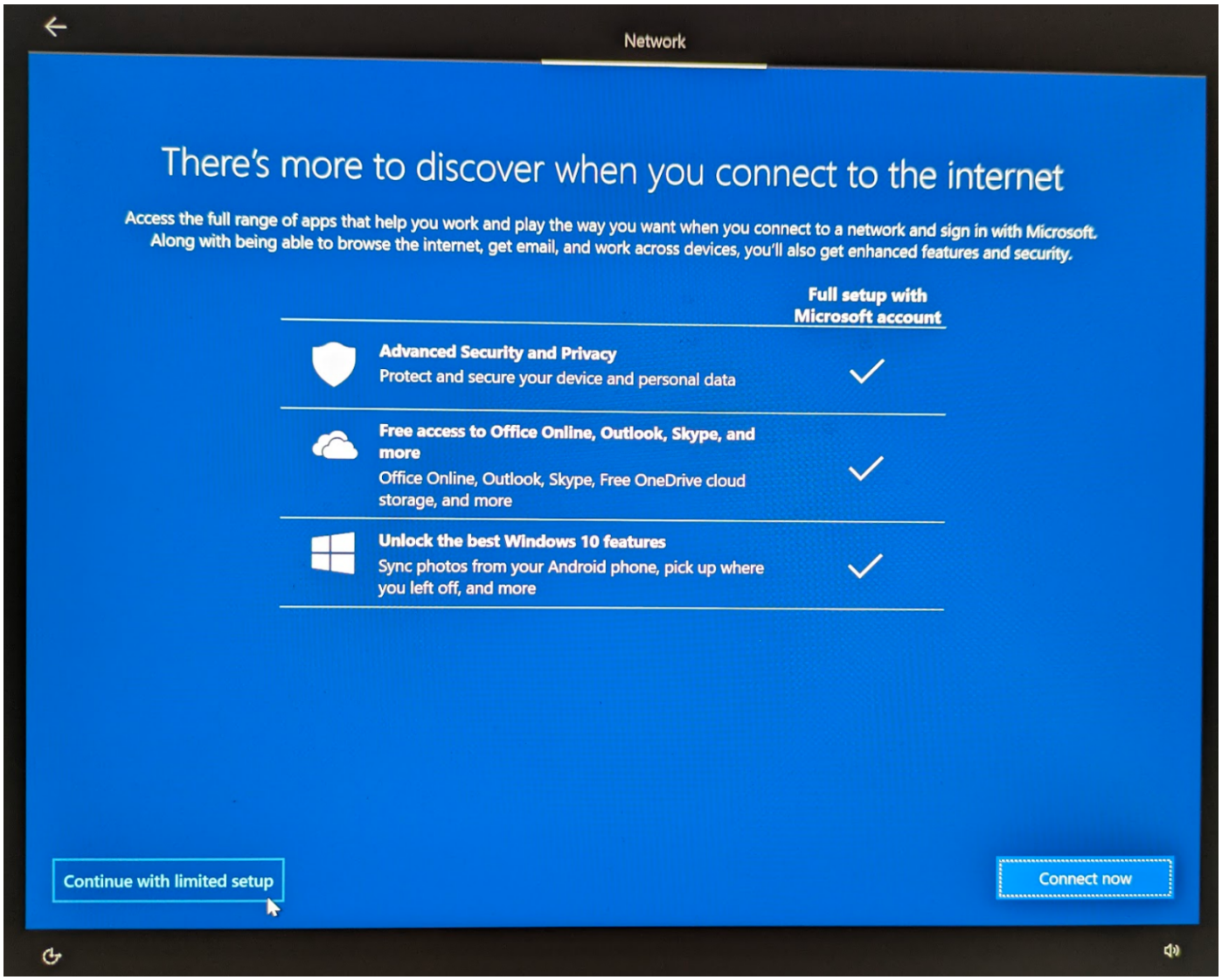
1. Turn on the workstation.
2. Press F10 repeatedly until the BIOS menu appears.
3. Enter the Workstation BIOS Password.
4. Navigate to "Advanced" > "Boot Options".
5. Check the box to enable “USB Storage Boot.”
6. Press F10 and follow the prompts to save changes and exit.
7. Following OS deployment, follow the same process to disable USB Storage Boot.

Then, using USB 2 with the Windows 10 image install the Windows OS

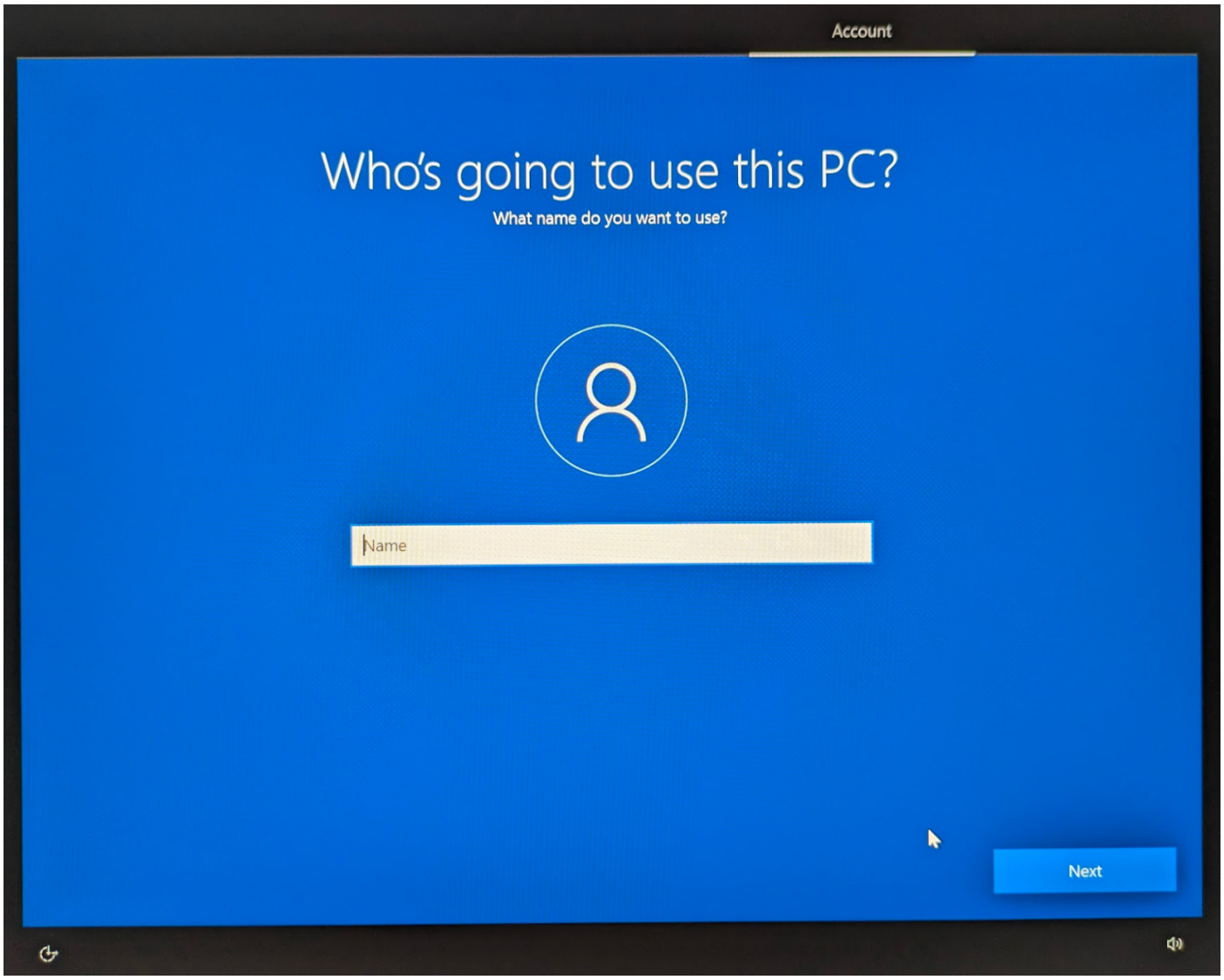
1. Insert USB and restart the computer. When it restarts you should see a Windows OS installation screen. Follow the instructions.

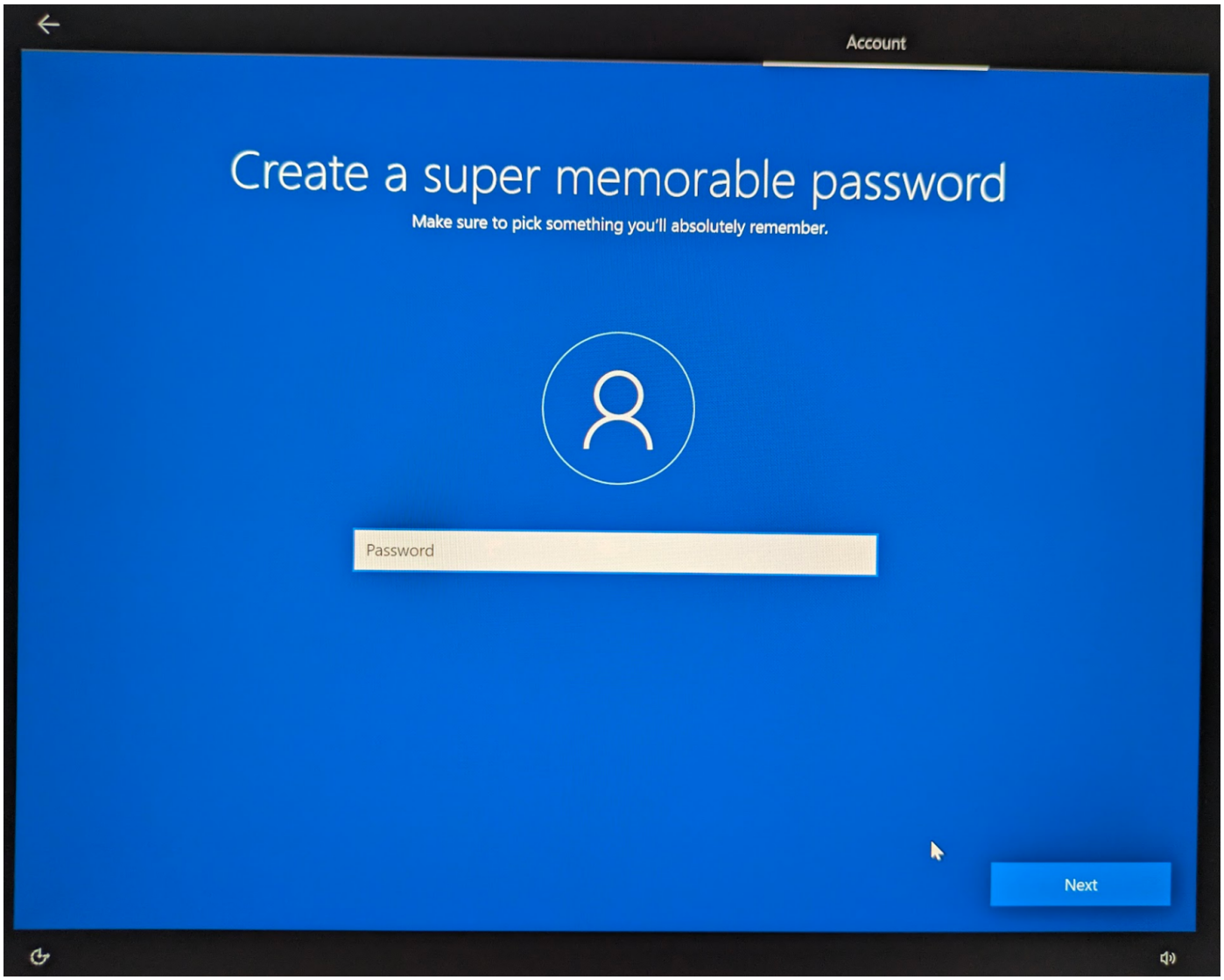
a. On the "Let's connect you to a network" screen select "I don't have internet" button in bottom left then "Continue with limited setup"



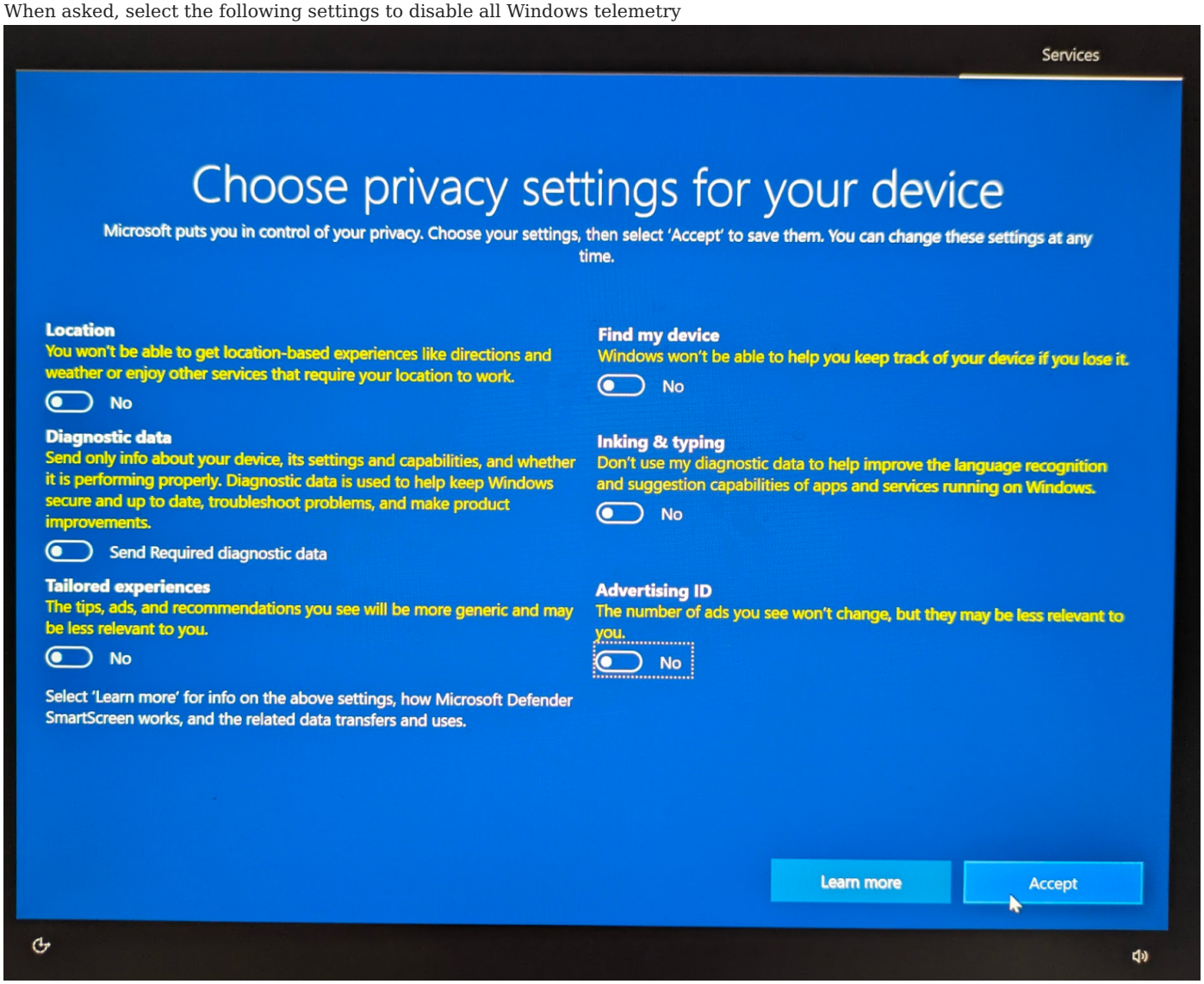


b. When it asks "Who's going to use this PC?" this will be the Windows Administrator Account. After providing an account name it will ask for a password.





c. When asked, select the following settings to disable all Windows telemetry



Then, disable USB boot.

1. Restart the workstation.
2. Press F10 repeatedly until the BIOS menu appears.
3. Enter the Workstation BIOS Password.
4. Navigate to "Advanced" > "Boot Options".
5. Check the box to disable “USB Storage Boot.”
6. Press "F10" and follow the prompts to save changes and exit.

2.21.4 Step 1: Prepare Updates on Internet Connected Computer

RCTab should run on a airgapped 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.

Preparing Windows Defender Offline Updates

- Navigate to the following website: https://www.microsoft.com/en-us/wdsi/definitions
- Go to the “Manually Download the Update” section. Download the update for your OS.
- Transfer the downloaded file to a USB drive and keep for later installation. As a note, all USB drives used for these purposes should be kept in a safe place in manufacturer recommended conditions.

Prepare LibreOffice Offline Installer

Download LibreOffice 7.6.4 installer `LibreOffice_7.6.4_Win_x86-64.msi` here. `SHA256`: `65678ac729cd0b545d14703879b601872d285c2934ae8d76452f7c2fb2c62d15`

Prepare XML Notepad Installer

XML Notepad can be used to find the Contest ID from `.xml` CVR files. The standalone installer can be downloaded as `XmlNotepadSetup.zip` from here. `SHA256`: `00cdfb1888cddf5507e691d610b3959428599317db97ac3de763e0e8940aef36`

2.21.5 Step 2: Applying Offline Updates

This section covers the configuration of the operating system and other software on the airgapped computer. Since the workstation is not connected to the internet, updates and security patches that were prepped in the previous step must be installed manually.

Note: Depending on the build of your Windows, the UI might be slightly different than described in the section below.

Install Xml Notepad

- Extract `XmlNotepadSetup.zip`
- Double-click the `XmlNotepadSetup.msi` to install Xml Notepad.

Install LibreOffice

- Double click the `LibreOffice_7.6.2_Win_x86-64.msi`
- Run through the installer with all default settings to install LibreOffice

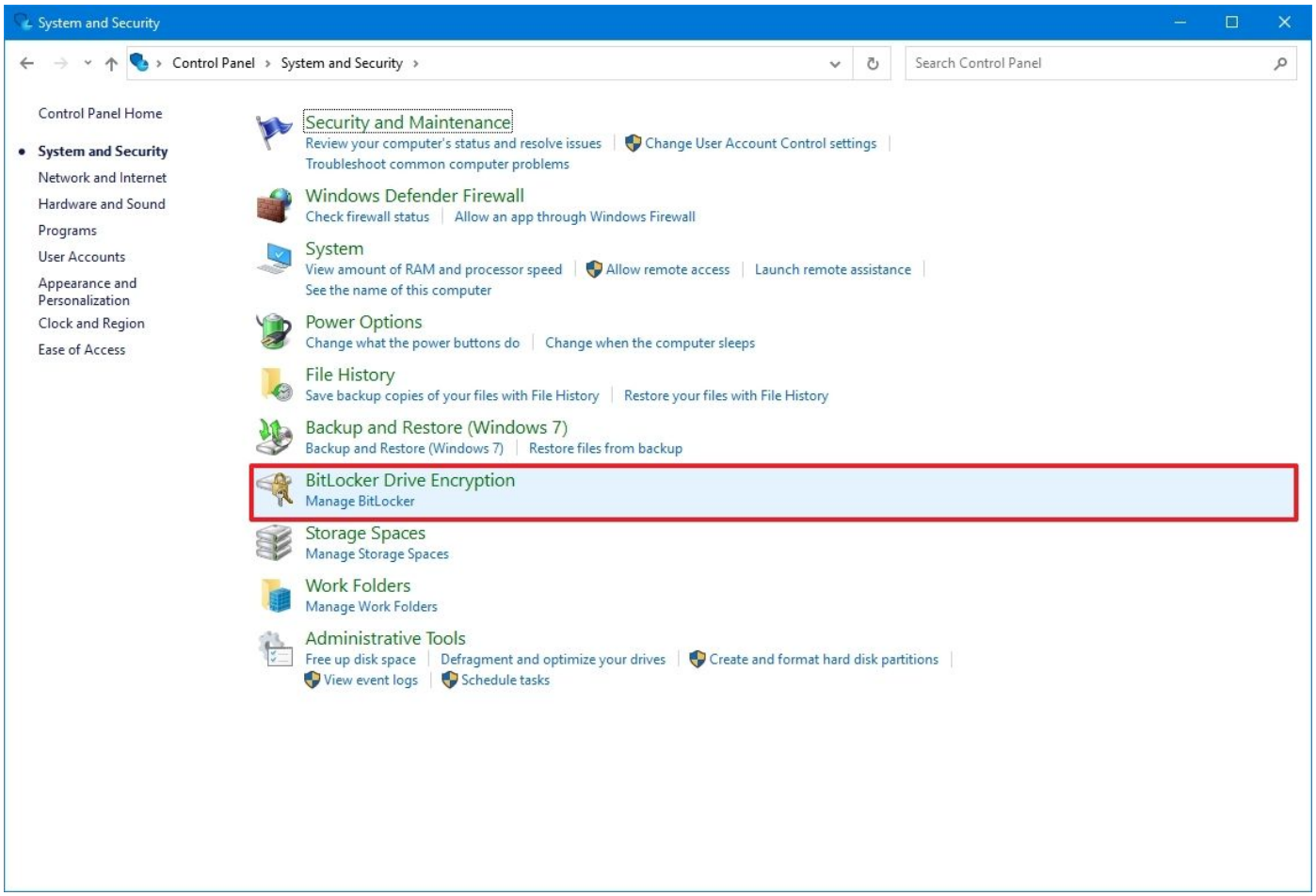
Windows Defender

- 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 by searching for “Virus and Threat Protection” in Windows Search, locate "Virus and threat protection updates" section. It will say "Up to date" if installation was successful.

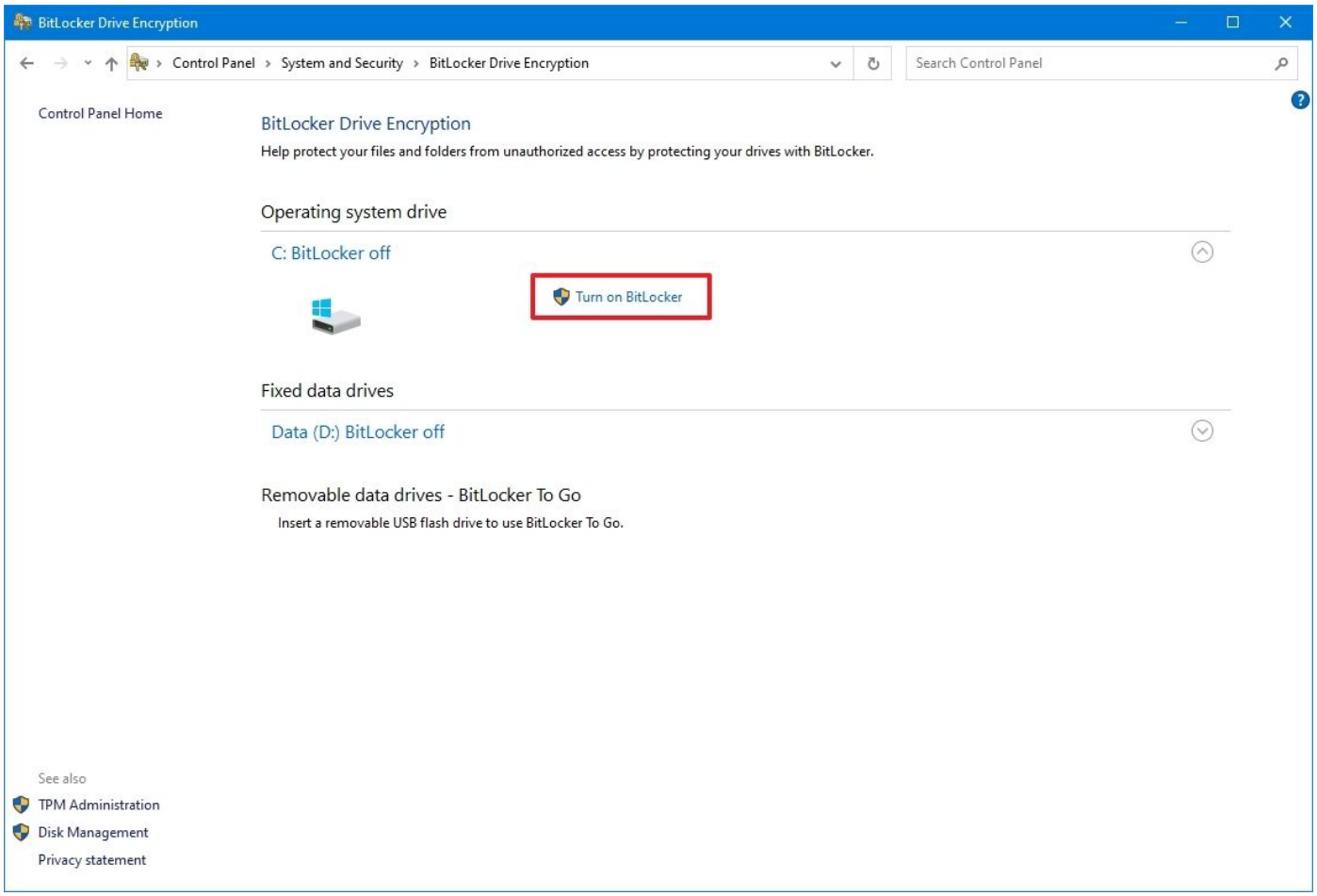
Device Encryption

Use the following steps to encrypt the hard drive with Windows Bitlocker

1. Open Start.
2. Search for "Control Panel" and click the top result to open the app.
3. Click on "System and Security".
4. Click on "BitLocker Drive Encryption".



5. Under the "Operating system drive" section, click the "Turn on BitLocker" option.

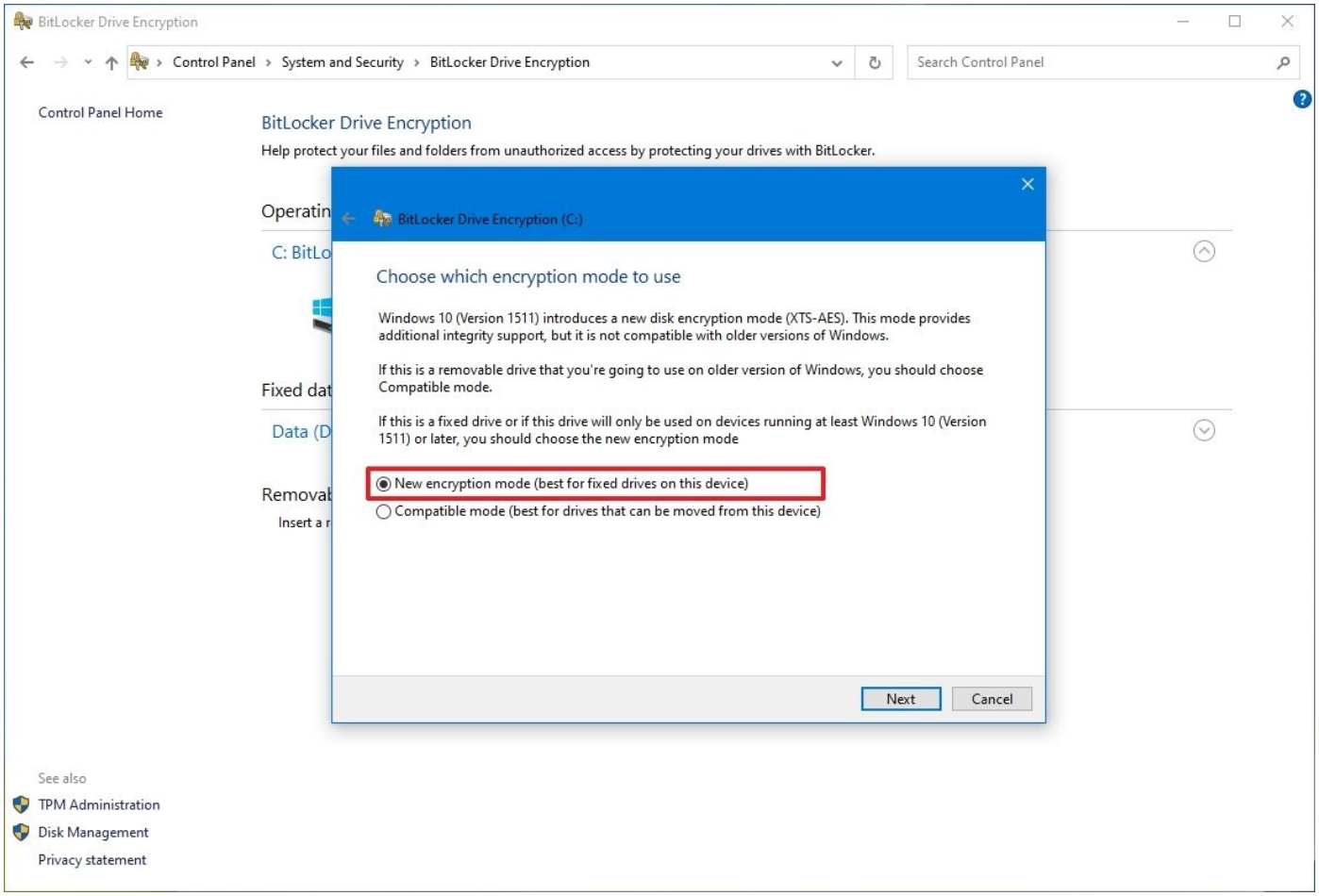


6. Select the option to save the recovery key.
7. Click the Next button.
8. Select how much the drive space to encrypt:

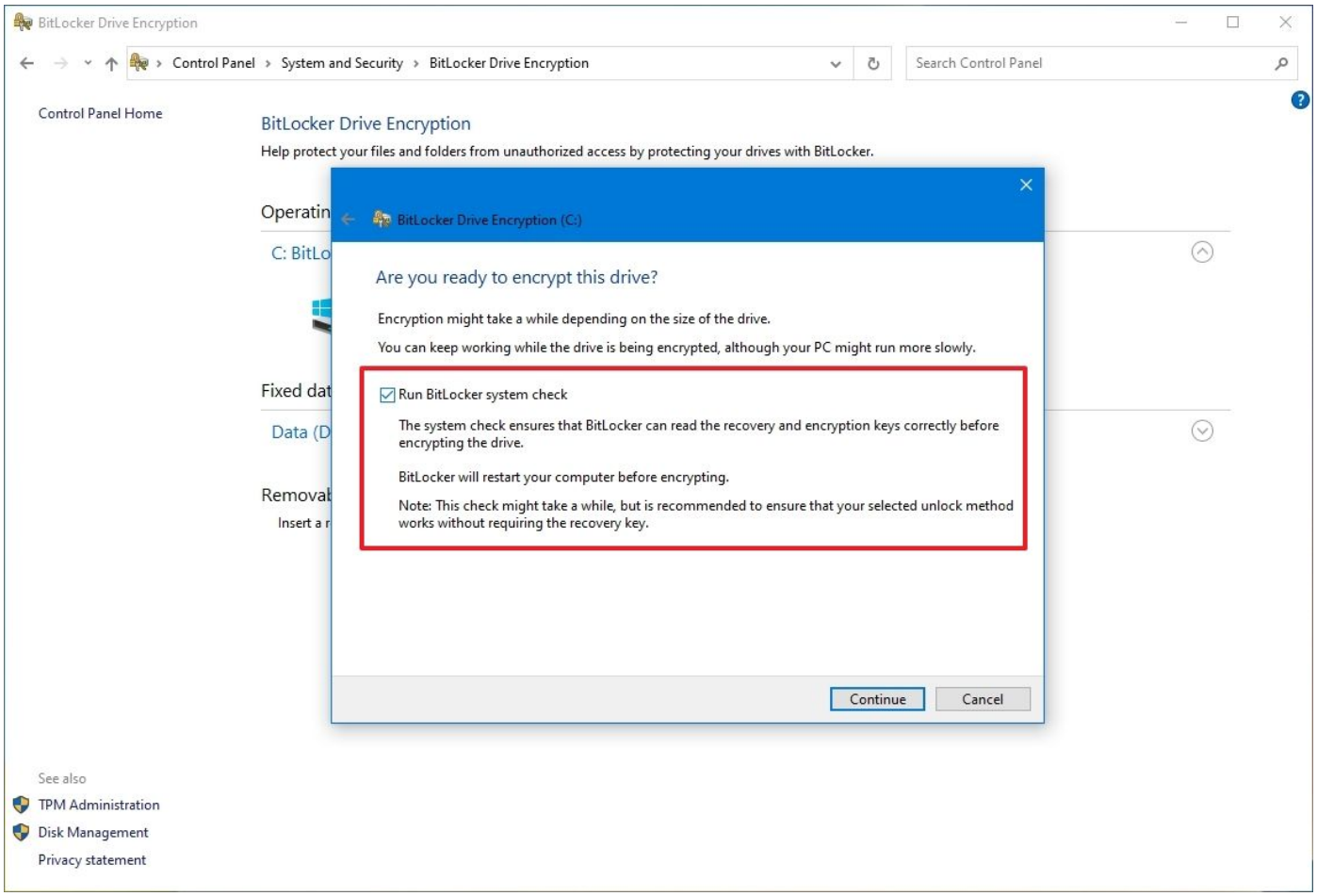
- Encrypt the entire drive (slower but best for PCs and drives already in use).

9. Choose between the two encryption options:

- New encryption mode (best for fixed drives on this device).



10. Click the Next button.
11. Check the "Run BitLocker system check" option.



12. Click the Continue button.
13. Click the Restart now button.

Physical Security

1. Users must physically seal all external ports on hardware where RCTab is installed, except:
   - Ports used for power supply,
   - Necessary external displays, and
   - One (1) USB port.
2. The user jurisdiction should implement a policy that employs tamper-evident and tamper-resistant seals to categorize ports based on their accessibility:

a. Ports that should never be accessed,

b. Ports that are unlikely to be accessed, and

c. Ports that 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. Manufacturer highly recommends the following hardening procedures to protect your standalone computer.

NOTE: Before hardening, 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.

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: "Select disable".

DISABLE SCREEN SAVER

This will ensure that users will always have to log in with their passwords.

1. Open the Control Panel.
2. Search “Screen Saver” and select “Change Screen Saver”
3. In the “Screen Saver” dropdown select “None”

DISABLE REMOTE ACCESS

1. Type "remote desktop settings" into the Windows search box.
2. Select "Remote Desktop Settings" to access the “Remote Desktop” settings
3. Ensure “Enable Remote Desktop” setting is set to "**off**"

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:

```
C:\Windows\System32\netsh.exe interface set interface <interface_name> disable
```

GLOBAL DISCLAIMER FOR NON-RCTAB 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.

2.22 Section 17 - System Test and Verification Specification

2.22.1 9.7 System Test and Verification Specification

The manufacturer shall provide test and verification specifications for: Development test specifications.

9.7.1 Development Test Specifications

9.7.1(A) THE MANUFACTURER SHALL DESCRIBE THE PLANS, PROCEDURES, AND DATA USED DURING SOFTWARE DEVELOPMENT AND SYSTEM INTEGRATION TO VERIFY SYSTEM LOGIC CORRECTNESS, DATA QUALITY, AND SECURITY

9.7.1(B) THIS DESCRIPTION SHALL INCLUDE: TEST IDENTIFICATION AND DESIGN, INCLUDING:

1. Test structure
2. Test sequence or progression; and
3. Test conditions

Test Structure

Included below is the basic template used to define every setting for each test set used to test 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.

Regression Test Overview

We have developed a suite of 68 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.

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 **Section 13 - Quality Assurance Plan**.

9.7.1(C) STANDARD TEST PROCEDURES, INCLUDING ANY ASSUMPTIONS OR CONSTRAINTS

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: https://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/hotfix/1.3.2. This is the location on GitHub where version 1.3.2 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 then further 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. 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 RCTab
6. Confirm that the configuration file properly loaded into RCTab by checking that the relevant fields are filled in
7. Click “Tabulation”
8. Click “Tabulate”
9. Wait for 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.

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.

9.7.1(D) SPECIAL PURPOSE TEST PROCEDURES INCLUDING ANY ASSUMPTIONS OR CONSTRAINTS

No special-purpose test procedures are followed.

9.7.1(E) 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 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.

9.7.1(F) 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.

9.7.1(G) CRITERIA FOR EVALUATING TEST RESULTS

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.

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 RCTab.

9.7.2 Test Specifications

The manufacturer shall provide specifications for verification and validation of overall software performance. These specifications shall cover:

1. Control and data input/output;

- Data input: Data used in 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 RCTab should be created using the guidelines and procedures outlined in **Section 18 - User Guide**.

2. Acceptance criteria;

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

3. 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 **Section 18 - User Guide**.

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

- Data quality is tested by ensuring that data used in 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.

5. 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 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.

6. Exception handling;

- The `invalid_params_test` and `invalid_sources_test` can be used to test exception handling in RCTab. When run through RCTab these tests will cause 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.
```

7. Security;

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

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

2.22.2 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 RCTab. This form is filled o
settings for the [TEST NAME]. Testers can compare the information in this form to the information in the configuration
the relevant folder in test_data to ensure that all settings are correct. Each setting selected in this form is a setting teste
RCTab has not yet created a Test Structure Form for every RCTab test, but will set up a form for all RCTab regression te
form. Questions about this form can be made to the Ranked Choice Voting Resource Center at info@rcvresources.org or
1-833-868-3728.

Contest Info

Contest Name*:	**Contest Date:**
Contest Jurisdiction:	**Contest Office:**
Rules Description:	

CVR Files

Provider*:	*Enter Vendor (ES&S, Hart, etc.)*
File Path*:	*Location of the cvr export file.*
First Vote Column Index*:	*Enter the Column where the First Vote is in export file.*
First Vote Row Index*:	*Enter the Row where the First Vote is in the file.*
ID Column Index:	*Enter the ID Column (if being used)*
Precinct Column Index:	*Enter the Precinct Column in the CVR expor*
Overvote Label:	
Undervote Label:	
Undeclared Write-In Label:	*User will define. Must match case and spelli file.*
Treat Blank as Undeclared WI:	*Circle One*

Candidates

Name*:	*Obtain a list of candidates from the EMS Sy the list of candidates to this form)*
Code:	*Enter the Code for each candidate (if being example: DDE*
Excluded:	*Check box if a candidate is not being counte tabulation*

Winning Rules

Winner Election Mode*:	Any user jurisdiction guidelines which ident rules for ranked choice voting elections
Maximum # of Ranked Candidates*:	attached to this form for easy reference. P RCTab User Guide for more information available selections that can be m

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*:				Any user jurisdiction guidelines which ident rules for ranked choice voting elections attached to this form for easy reference. P
Most Common Threshold:				RCTab User Guide for more information available selections that can be m
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 ra check Unlimited if permitted.
Exhaust on multi ranks for same candidate:	Y	N	(Circle One)	Check if multiple ranks for the same candid permitted.

Output

Output Directory (where should results file go on PC?):				
Tabulate by Precinct:	Y	N	(Circle One)	**Generate CDF JSON:** Y N (Circle On

Configuration File

Name:

Date of Test: **Passed?:** Y N (Circle On

Name of tester(s):

Name 1 **Name 2**

‡Disclaimer: The Hare Quota tabulation option in the RCTab software has not been thoroughly tested in a controlled tes
environment. Do not attempt to implement this option without first testing in a non-operational environment. Please con
Choice Voting Resource Center for additional information.

2.22.3 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 RCTab. This form is filled o
settings for the [TEST NAME]. Testers can compare the information in this form to the information in the configuration
the relevant folder in test_data to ensure that all settings are correct. Each setting selected in this form is a setting teste
RCTab has not yet created a Test Structure Form for every RCTab test, but will set up a form for all RCTab regression te
form. Questions about this form can be made to the Ranked Choice Voting Resource Center at info@rcvresources.org or
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, Hart, 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 export file.*
First Vote Row Index*:	2	*Enter the Row where the First Vote is in the file.*
ID Column Index:	Leave blank	*Enter the ID Column (if being used)*
Precinct Column Index:	2	*Enter the Precinct Column in the CVR expor*
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 spelli file.*
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 Sy the list of candidates to this form)*
Code:		*Enter the Code for each candidate (if being example: DDE*
Excluded:		*Check box if a candidate is not being counte tabulation*

Winning Rules

Winner Election Mode*:	singleWinnerMajority	Any user jurisdiction guidelines which ident rules for ranked choice voting elections
Maximum # of Ranked Candidates*:	15	attached to this form for easy reference. P RCTab User Guide for more information
	0	available selections that can be m

Minimum Vote Threshold:		
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`	Any user jurisdiction guidelines which ident rules for ranked choice voting elections attached to this form for easy reference. P
Most Common Threshold:	`N/A`	RCTab User Guide for more information available selections that can be m
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 ra check Unlimited if permitted.
Exhaust on multi ranks for same candidate:	Y **N** (Circle One)	Check if multiple ranks for the same candid 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 Or

Configuration File `2015_portland_mayor_config.json`

Name:

Date of Test: **Passed?:** Y N (Circle On

Name of tester(s):

Name 1 **Name 2**

‡Disclaimer: The Hare Quota tabulation option in the RCTab software has not been thoroughly tested in a controlled tes
environment. Do not attempt to implement this option without first testing in a non-operational environment. Please con
Choice Voting Resource Center for additional information.

List of 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.

2.22.4 List of Tabulator Tests

Name of Test	Name of Test Folder	Description of test	Purpose of Test
RCVRC & Bright Spots name for regression test	Test folder name in: 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 **Section 06 - 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;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
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	Control and data input/ output; Processing accuracy; Ballot interpretation logic;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		functionality of the "multi-winner allow multiple winners per round" functionality.	
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 has been updated but requires additional testing. 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	Control and data input/ output; Processing accuracy; Ballot interpretation logic;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		with ES&S CVR files.	
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	Control and data input/ output; Processing accuracy; Ballot

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		ES&S CVR files, as well as RCTab's ability to properly count real-life RCV elections.	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	Control and data input/ output; Processing accuracy; Ballot

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		count on that data.	interpretation logic;
Dominion - Multi-File	`dominion_multi_file`	Tests the ability of RCTab to read and process Dominion CVR data spread across multiple CVR files 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 inactivating ballot after encountering duplicate ranking of same candidate	`duplicate_test`	Tests the ability of RCTab to properly detect and inactivate a ballot due to the same candidate being ranked multiple times on that ballot.	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	Control and data input/

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		RCTab to read and process Hart CVR data	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	Control and data input/ output; Processing accuracy; Ballot interpretation logic;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		multiple winners per round" functionality.	
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 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;
No candidates meet minimum vote threshold	`no_candidates_meet_minimum`	Tests the ability of RCTab to provide an accurate error	Exception handling.

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		message if all candidates fall below the value set in the Minimum Vote Threshold field.	
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	Control and data input/ output; Processing accuracy; Ballot interpretation logic;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		user input. It is included in the "sample_input" folder of RCTab installation folder as an additional test of RCTab.	
Sequential with batch	`sequential_with_batch`	Tests the ability of RCTab to properly batch eliminate candidates in a multi-pass IRV tabulation scenario.	
Sequential with continue until two	`sequential_with_continue_until_two`	Tests the ability of RCTab to eliminate down to the final two candidates in a multi-pass IRV tabulation scenario.	
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	Control and data input/ output; Processing accuracy; Ballot

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		the ability of RCTab to export CVR data in the CDF.	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"	Control and data input/ output; Processing accuracy; Ballot

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		setting when ballots exhaust after multiple skipped rankings. Test the ability of RCTab to export CVR data in the CDF.	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	Control and data input/ output; Processing accuracy;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		data files in XML format. Test the ability of RCTab to export CVR data in the CDF.	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.	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;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
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 previousRoundCountsThenRandom	`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

Name of Test	Name of Test Folder	Description of test	Purpose of Test
			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	`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	Control and data input/ output; Processing accuracy; Ballot interpretation logic;

Name of Test	Name of Test Folder	Description of test	Purpose of Test
		exception by not awarding a win to an undeclared write-in candidate in a single-winner contest.	

2.22.5 List of Security Tests

Name of Test	Description of test	Purpose of Test
RCVRC & Bright Spots name for regression test	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 - System Security Specifications. Exception handling tests are directly identified below.
Succeeds using the default, valid files	Tests that properly signed Hart CVRs succeed cryptographic validation	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Verification fails when the signature is incorrect	When the cryptographic signature of the Hart CVR is incorrect, confirm that it successfully throws an exception	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Exception is thrown when the data file is modified	Confirms that if the signed Hart CVR file itself is modified that signature validation should fail	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Succeeds when data file is in a different folder (but has the right filename)	Hart includes a path in the data that is signed. Since CVRs will be moved from their original location to the RCTab machine, we make sure that this doesn’t affect signature verification.	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Exception is thrown when the filenames differ	Though files can move, we require that file names stay the same.	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Exception is thrown when the file is signed with an unsupported key	When the public key is not a valid public key, ensure that signature verification fails	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly
Ensure FIPS Compliance check is run	Tests that Java Security Providers are properly setup to ensure FIPS compliance	Ensuring cryptographic validation of signed Hart CVRs is implemented correctly

2.23 Section 18 - User Guide

Users should ensure instructions below have been followed and completed prior to operating RCTab:

- **Section 05 - Acceptance Test Procedures**
- **Section 16 - System Hardening Procedures - Windows OS**
- **Section 22 - Installation Instructions for Windows OS**
- **Section 23 - HashCode Instructions - Windows OS**
- **Section 25 - Configuration File Parameters**

Any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team with no less than two trained personnel. This document describes all interfaces and options in the RCTab software.

2.23.1 Launching RCTab

The manufacturer recommends RCTab be installed as part of the pre-election preparation process. In order to determine the appropriate launch procedure for a jurisdiction, users should consider the maximum possible number of votes that could occur in the event all eligible voters presented themselves to vote. Jurisdictions should, as part of their pre-election procedure, launch RCTab according to the relevant launch instructions described below to ensure they launch it in accordance with the below requirements. The manufacturer is available for support with this process. For acceptance testing and L&A procedures to set up and use of RCTab, see:

- **Section 05 - Acceptance Test Procedures**
- **Section 11 - L&A Testing**

Contests with fewer than 1,000,000 votes

To Launch:

1. Navigate to the rcv folder created when you unzipped RCTab.
2. Open the bin folder
3. Right-click on the `rcv.bat` file. Click "Run as Administrator". If a "Windows protected your PC" window pops up click "More Info" then click the "Run anyway" button. Enter the administrator password

Contests with more than 1,000,000 votes

1. Open a Command Prompt by navigating to the start menu and typing in Command Prompt.
2. Press enter to launch Command Prompt.
3. Change the current directory to the `rcv` folder created when you unzipped RCTab.
4. First, type in cd (note there should be a space after cd).
5. Using File Explorer navigate to the folder where RCTab is installed.
6. Double-click on the `rctab_v1.3.0_windows` folder
7. Click and drag the `rcv` folder over to the command prompt window
8. Your command prompt will now read something like:

a. `cd C:\RCTab\rctab_v1.3.0_windows\rcv`

9. Press enter
10. Launch the tabulator by entering the following command:

a. `.\bin\java -mx30G -p .\app -m network.brightspots.rcv/network.brightspots.rcv.Main`.

2.23.2 Prepping CVRs for use with RCTab Software

All exports of CVR records for use with RCTab and migration of CVR records for use with RCTab should follow CVR procedures required by the jurisdiction. Users must keep track of the path to each file that is needed to tabulate the RCV contests. This will be needed when configuring the RCTab software.

2.23.3 Setting up a configuration file for RCTab

All settings in the RCTab software are described in technical detail in **Section 25 - Configuration File Parameters**, including brief descriptions of how options impact the operational validity of other options. This document will describe how to set up a configuration for a contest using your jurisdiction's ranked choice voting rules. This guide also includes screenshots of the interfaces described. The values a user inputs into any of these fields depend upon the relevant laws and regulations in place in their jurisdiction, as well as the voting system vendor used to produce cast-vote records for their elections. Users must understand the requirements of their laws, regulations, and vendor CVR data in order to fill out these fields accurately for their needs.

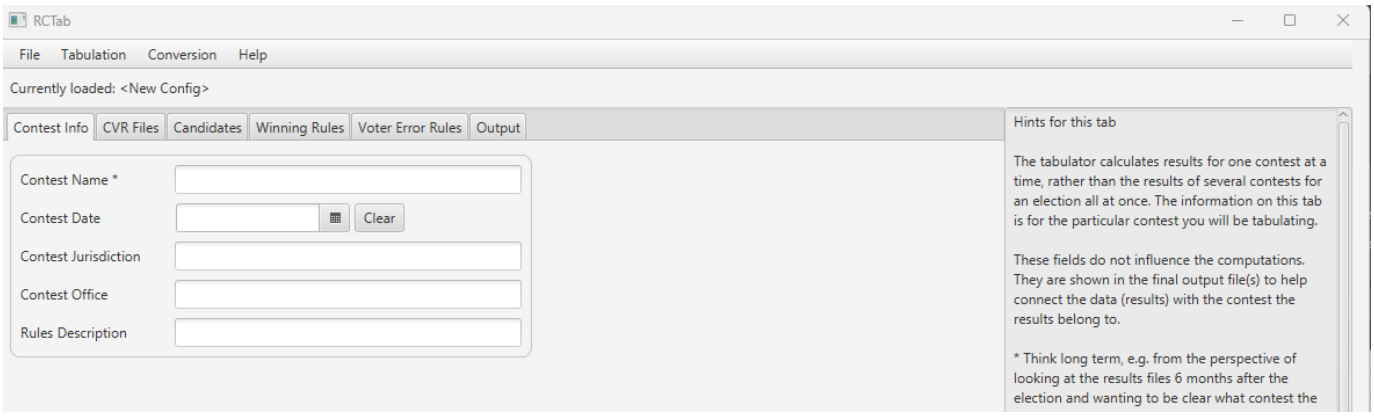
RCTab is organized into tabs: the Contest Info Tab, the CVR Files Tab, the Candidates Tab, the Winning Rules Tab, the Voter Error Rules Tab, and the Output Tab. These tabs each include a set of fields that users fill out - some fields are always required, some fields are required based on previous input, and some fields are always optional. This guide will take the user through basic descriptions of each of these tabs and each of these fields. Additional technical information about these fields can be found in **Section 25 - Configuration File Parameters**. As users navigate through each tab they are building a configuration `.json` file which must be saved when complete. If RCTab is closed and the configuration is not saved, no information will be stored. The next time RCTab is opened all fields will be blank. Once a configuration is saved, it can be used by RCTab to process RCV election results according to the rules laid out in that configuration, provided all rules necessary are filled out.

Each individual contest run through RCTab requires its own configuration with contest-specific information such as the contest name and candidate names. This guide will run through setting up the winning rules and voter error rules requirements that will apply in your election and will discuss procedures for how to properly fill out other fields in the software. Operation of RCTab must be conducted by a team of no less than two trained personnel. Note: Including a % symbol anywhere in a configuration file results in a crash of the software. Do not use % symbols in any portion of a configuration file, including settings such as file paths and candidate names.

The following guide describes how to operate RCTab. See also **Section 11 - L&A Testing** document for additional detail on system operations.

Contest Info Tab

Below is what the contest info tab looks like when a user first navigates to it. This is also the first screen a user will see upon successfully launching RCTab. Contest info fields are in the middle of the screen. The black box at the bottom of the screen is called the operation log box. It sends the user messages about the tabulation process, any errors encountered in using a configuration file, any errors in tabulation, and other software errors. Information in this panel is saved to the `rcv_0.log file`. Users cannot turn this feature off while using the RCTab software. The panel on the right side of the document is the “Hints'' tab which includes any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team with no less than two trained personnel.



These settings appear on the "Contest Info" tab in RCTab.

- Contest Name
- Contest Date
- Contest Jurisdiction
- Contest Office
- Rules Description

The tabulator calculates results for one contest at a time, rather than the results of several contests for an election all at once. The information on this tab is for the particular contest you will be tabulating.

These fields do not influence the computations. Contest Name, Contest Date, Contest Jurisdiction, and Contest Office are shown in the final output file(s) to help connect the data (results) with the contest the results belong to.

- Think long term, e.g. from the perspective of looking at the results files 6 months after the election and wanting to be clear what contest the results belong to.
- You may find it helpful to revisit this tab once you have done a few test runs and see what the output looks like.

Contest Name (required): Enter a name to identify it.

- Examples: City Council 2018, Board of Election Ward 13 2017, Mayor, Referendum 289b

Contest Date (optional): The date on which the election for this contest was run.

- Clear permits user to clear this field
- The calendar button allows a user to navigate through a calendar to select the date of the contest

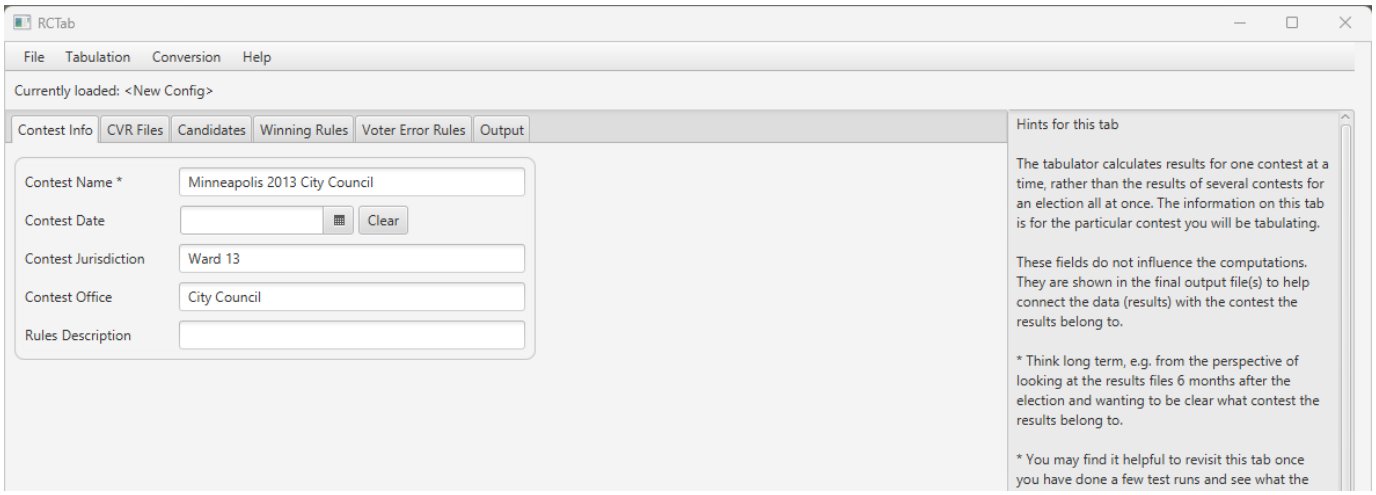
Contest Jurisdiction (optional): E.g.: Minneapolis, Eastpointe

- Whether this is helpful may depend on what you put into the Contest Name field

Contest Office (optional): E.g.: Mayor, County Clerk

- Whether this is helpful may depend on what you put into the Contest Name field

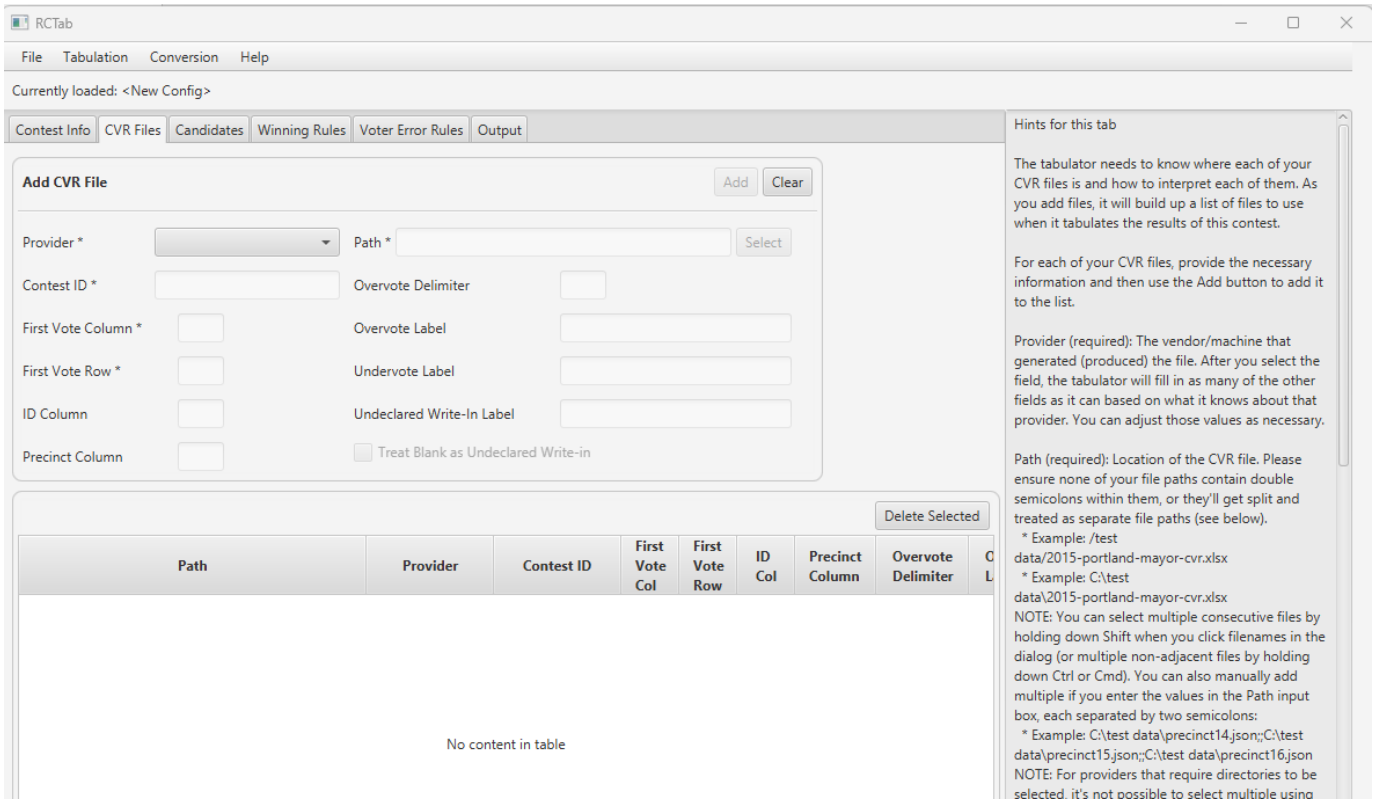
Rules Description (optional): What short description of this configuration would help you remember in, say, six months what election this specific rule configuration is for? This option’s use impacts no other options. It is included in configuration `.json` files and audit `.log` files.



Example Completed Contest Info tab

CVR Files Options

This is what the CVR Files tab looks like when a user first navigates to it:



The tabulator needs a configuration file laying out where each of its user's CVR files for the contest to be tabulated are and how to interpret each of them. Only add files to the configuration that contain data for the contest you are tabulating. CVRs that do not include data for the contest to be tabulated will cause tabulation to fail. Fields used to describe CVRs are on the top half of the tab, while the table on the bottom half of the tab will display each CVR file you have added to your configuration file so far. All information in this tab directs RCTab on where to find files and how to read those files, it does not actually pull in any vote information. Vote information is only used after a user clicks the "Tabulate" button under "Tabulation" as described later in this guide.

For each of your CVR files, provide the necessary information and then use the Add button to add it to the list.

Provider (required): The vendor/machine that generated (produced) the file. After you select the field, the tabulator will fill in as many of the other fields as it can, based on what it knows about that provider. You can adjust those values as necessary. Different options are active for different providers. See below for a break-down of each field and what is required:

Path (required)**:** Location of the CVR file.

- Example: `/Users/test data/2015-portland-mayor-cvr.xlsx`
- Select allows a user to navigate using Windows Explorer to select a location.

Contest ID: Some CVRs assign an ID label to each contest in the CVR. The tabulator needs to know which contest is being tabulated when multiple contests are included in one CVR. Enter the ID of the contest being tabulated in this field.

First Vote Column: the column where the first vote record is.

First Vote Row: the row where the first vote record is.

ID Column: The column the IDs are in. Not all CVR files contain an ID column.

Precinct Column: The column that contains the precinct.

Overvote Delimiter (optional, but must be blank if "Overvote Label" is provided)**:** If using a CVR in ES&S style, overvotes can be reflected in a CVR by displaying all candidates marked at a ranking. Those candidate names will be differentiated from each other by a delimiter, something like a vertical bar `|` or a slash `/`. If your overvotes are delimited like this, enter the delimiter used in this field. Note: that ES&S files may include only the label "overvote" and no additional information, in which case the "Overvote Label" field should be used instead.

Overvote Label (optional, but must be blank if Overvote Rule is)**:** Some CDF and ES&S CVRs use a particular word/phrase to indicate an overvote.

Undervote Label (optional)**:** Some ES&S CVRs use a particular word/phrase to indicate an undervote.

Undeclared Write-in Label (optional)**:** Some CVRs use a particular word/phrase to indicate a write-in.

Treat Blank as Undeclared Write-in (optional for ES&S)**:** When checked, the tabulator will interpret blank cells in this ES&S CVR as votes for undeclared write-ins.

Add: Adds CVR to the configuration file. Impacts no other option. Operation impacted by whether required fields (depending on Provider selected) are filled in.

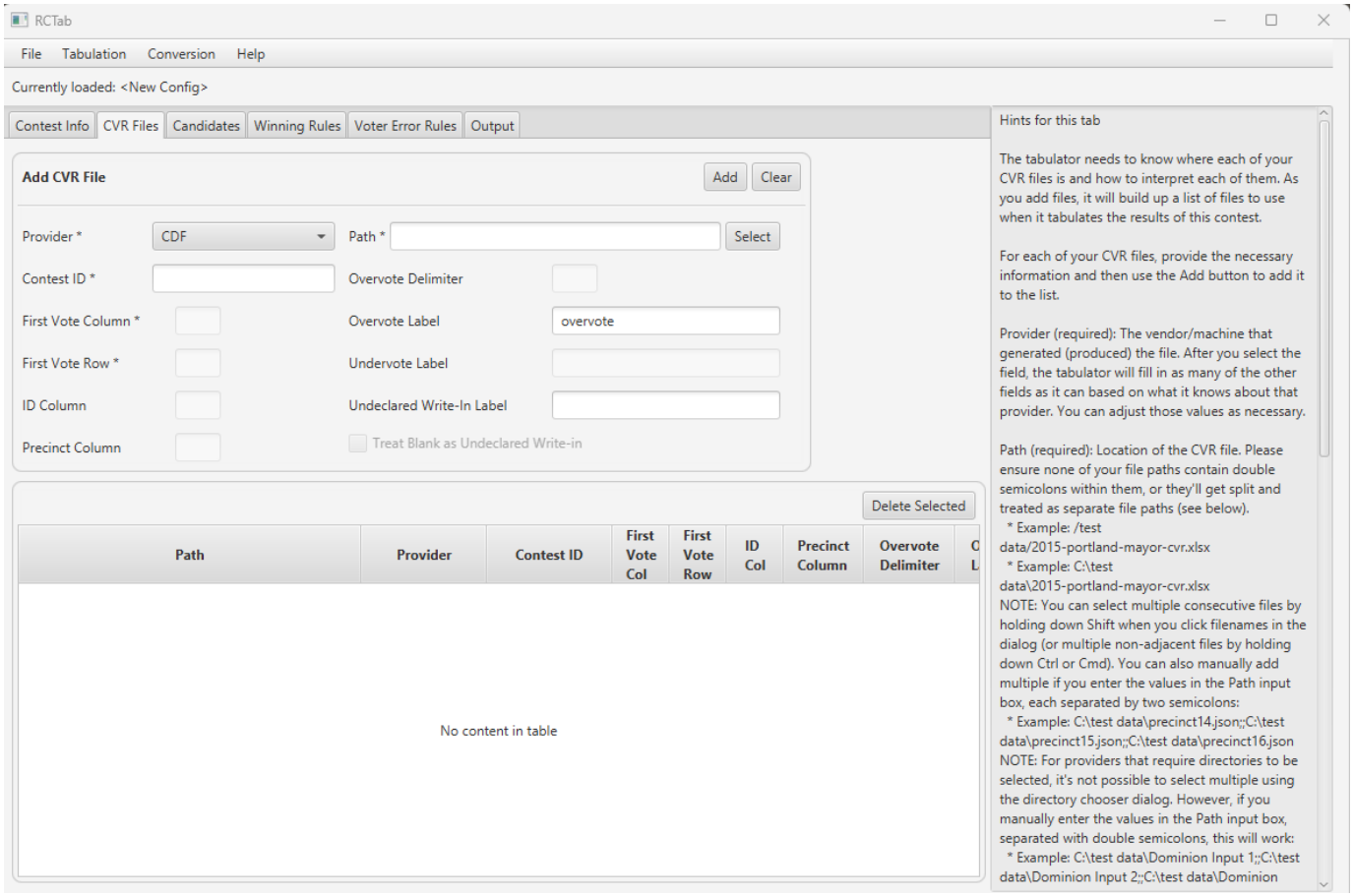
Clear: Clears all fillable values in CVR Files tab above the CVR Files table.

Delete Selected: Deletes CVR file information listed in the CVR table. Impacted by and impacts no options.

This guide will now briefly show which provider settings permit users to edit which additional settings.

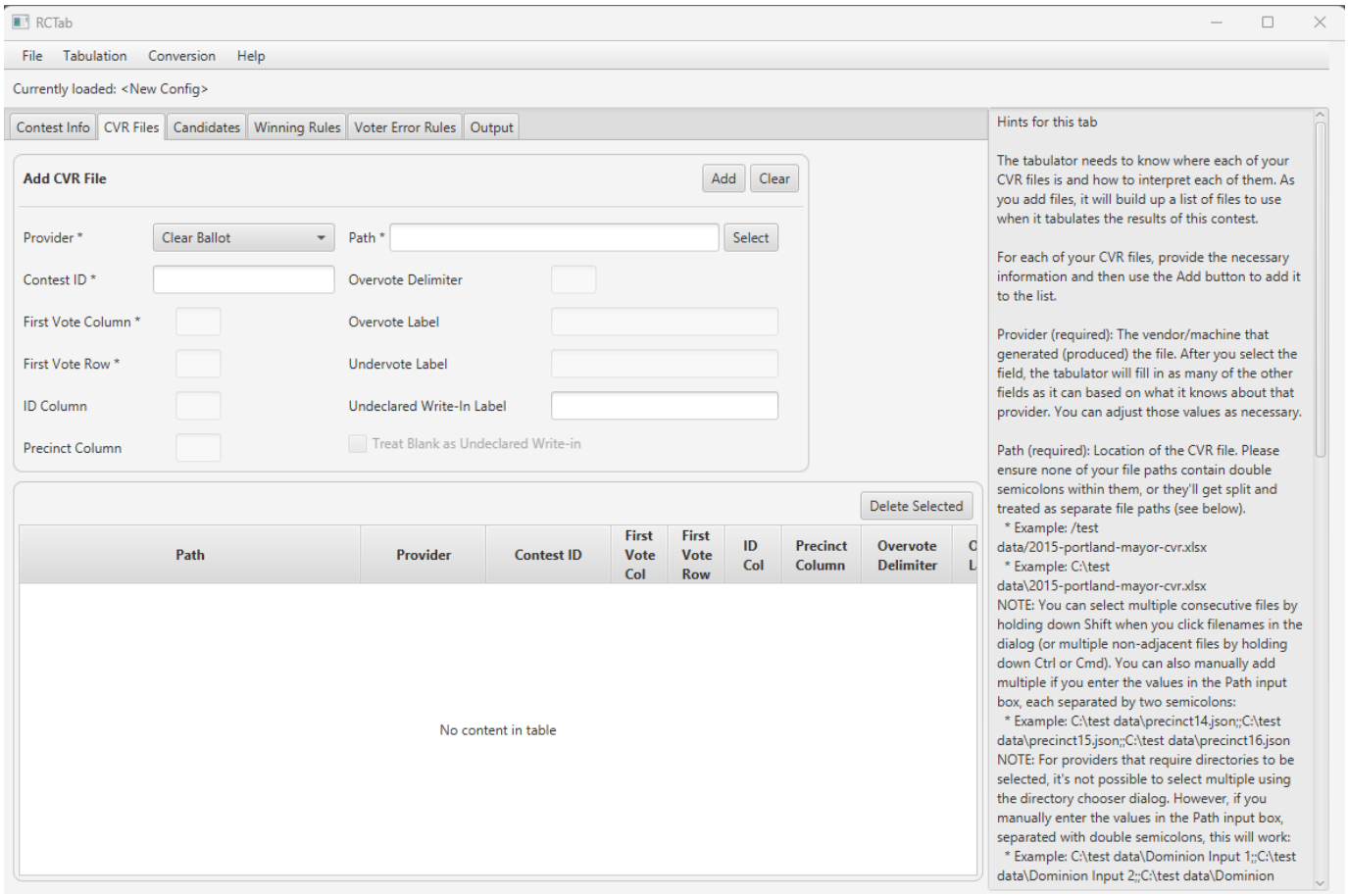
- Provider options:
- CDF
- Clear Ballot
- Dominion
- ES&S
- Hart

CDF



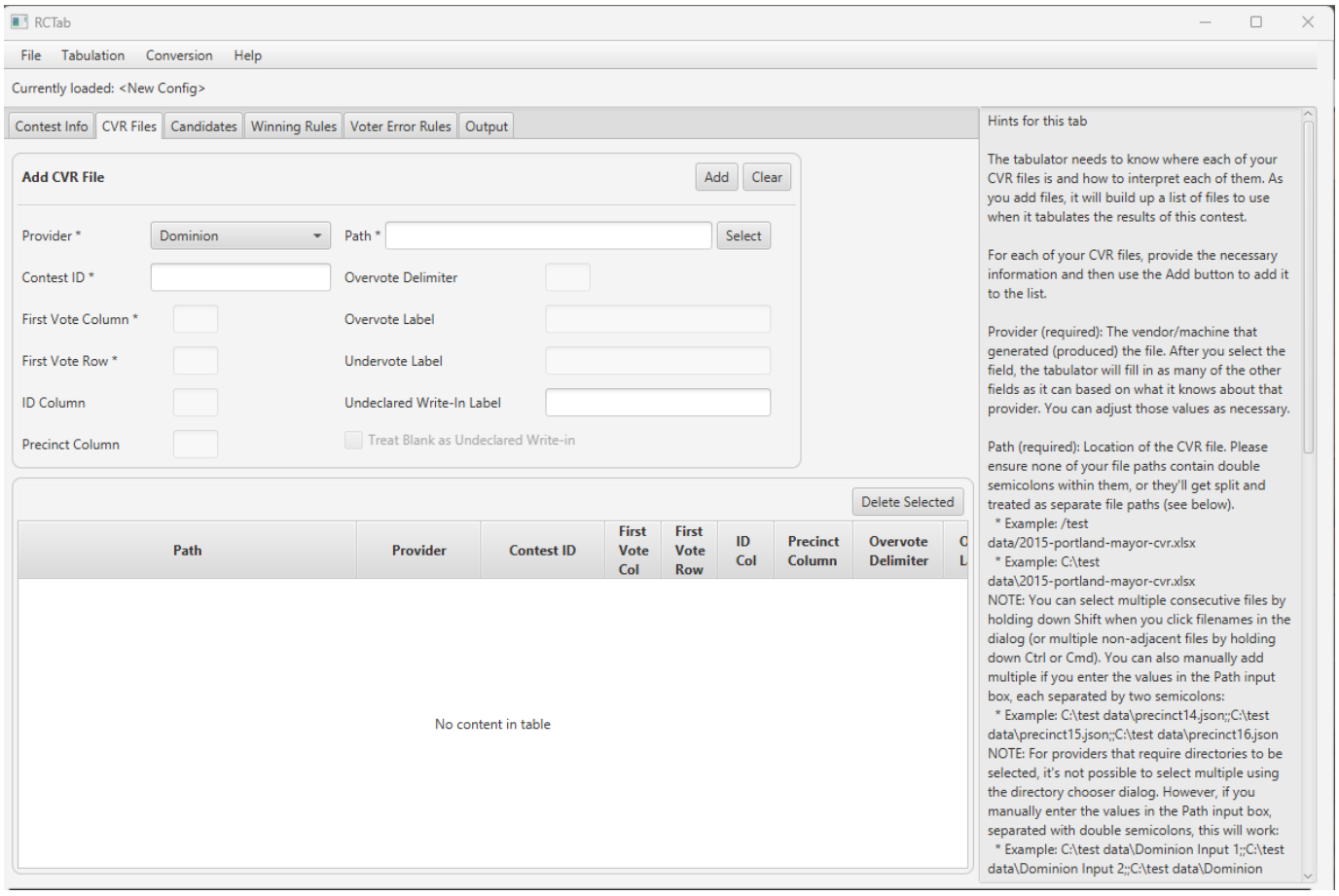
Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Overvote Label (Default value: overvote)	ID Column
Undeclared write-in label	Precinct Column
	Overvote Delimiter
	Undervote Label
	Treat Blank as Undeclared Write-In

CLEAR BALLOT



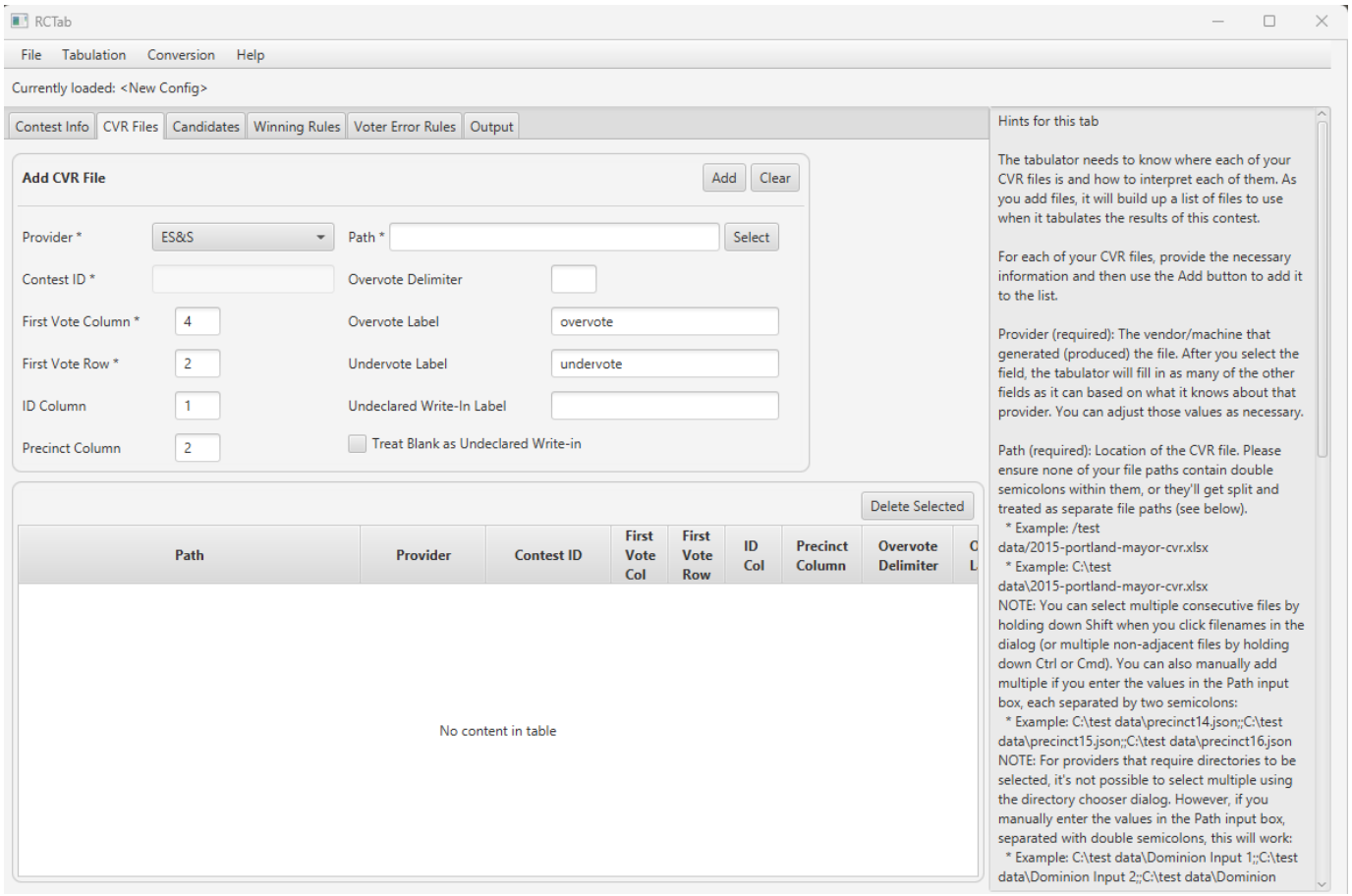
Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

DOMINION



Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

ES&S



Permits user to edit these options:	Does not permit user to edit these options:
Path	Contest ID
First Vote Column (Default Value: 4)	
First Vote Row (Default Value: 2)	
ID Column (Default Value: 1)	
Precinct Column (Default Value: 2)	
Overvote Delimiter	
Overvote Label (Default Value: overvote)	
Undervote Label (Default Value: undervote)	
Undeclared Write-In Label	
Treat Blank as Undeclared Write-In	

It is assumed that the ES&S export uses the default settings so below are the ES&S default settings. The user will only need to enter the path, but all other default settings are included. If the default settings are not used in the ES&S export file, the user must determine the values to be used. If defaults are not being used, users should confirm the proper values for First Vote Column, First Vote Row, ID Column, Precinct Column, Overvote Label, Undervote Label, and Undeclared Write-In Label by referencing the CVRs files to be used in the Tabulation.

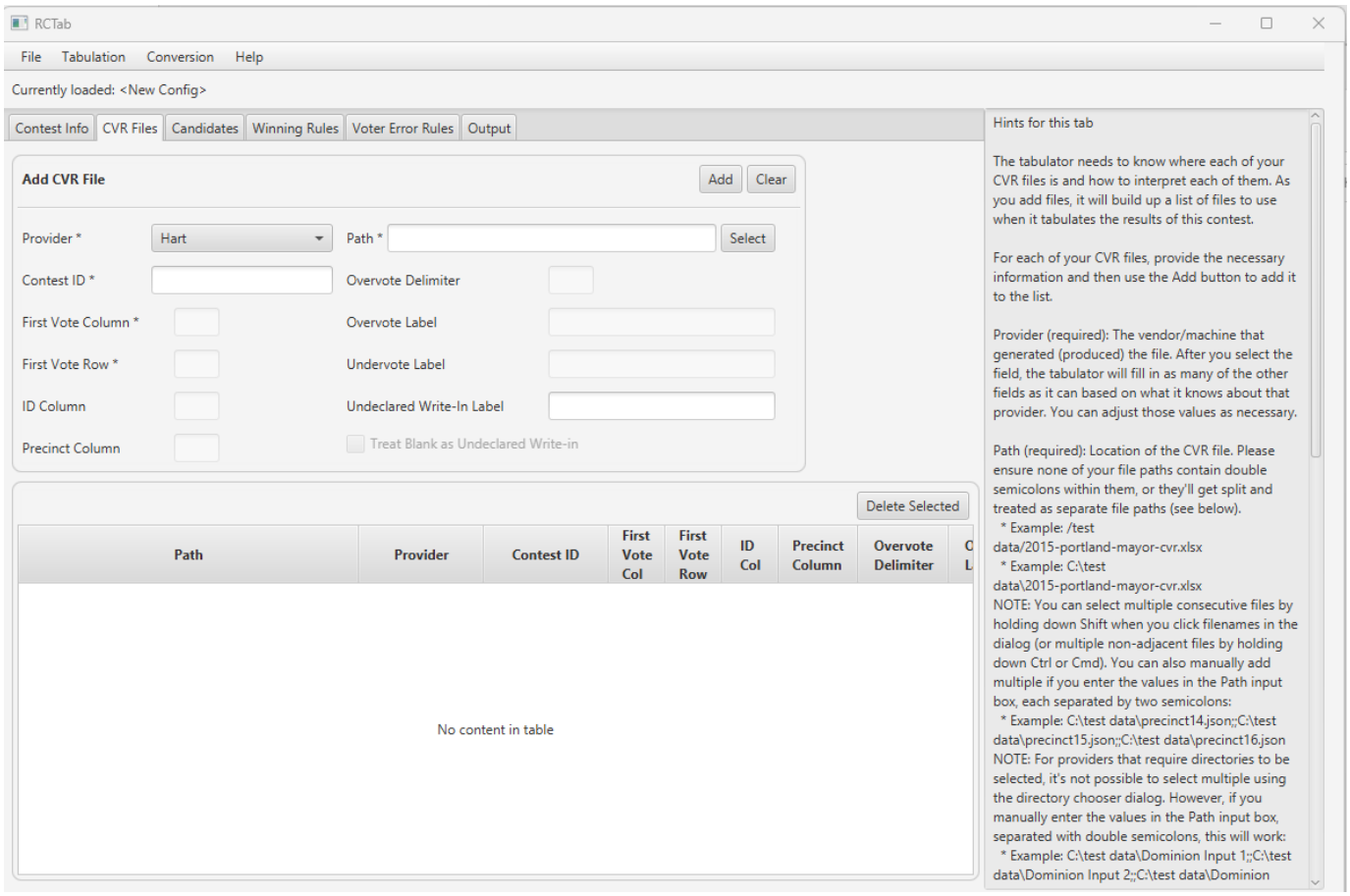
The First Vote Column location depends on the information users choose to export in their CVR exports from ES&S's system. ES&S systems can include individual CVR ID information, Precinct information, and ballot style label information in the first

three columns of a CVR, as shown in the below screenshot. Column A (aka Column 1) has CVR ID numbers; Column B (Column 2) has Precinct information, Column C (3) has ballot style information, and Column D (4) includes the first ranking in the RCV contest in this CVR. This is the standard CVR layout for ES&S. More information is available from ES&S.

A	B	C	D
Cast Vote Record	Precinct	Ballot Style	Rep. to Congress 1st Choice District 2
1	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
2	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
3	Fayette	CAN Ballot Style 130	DEM Golden, Jared F.
4	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
6	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
8	Fayette	CAN Ballot Style 130	Hoar, William R.S.
10	Fayette	CAN Ballot Style 130	undervote
15	Fayette	CAN Ballot Style 130	DEM Golden, Jared F.
17	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
19	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
20	Fayette	CAN Ballot Style 130	Hoar, William R.S.

CVR exports from ES&S contain columns for each ranking in an RCV contest. See the above screenshot for an example. This screenshot includes the five rankings voters had in the contest, arranged in sequential order. This CVR export included only data for the one RCV contest to be run. The setting “Maximum Number of Candidates That Can Be Ranked” (covered below in Winning Rules) instructs RCTab how many columns to process when running the round-by-round count. If that setting is set to five and the First Vote Column setting is set to four (Column D), RCTab will process this CVR starting at Column D/4 (column number 4) and continue through columns E/5, F/6, G/7, and H/8 - five columns for the five rankings voters have.

HART

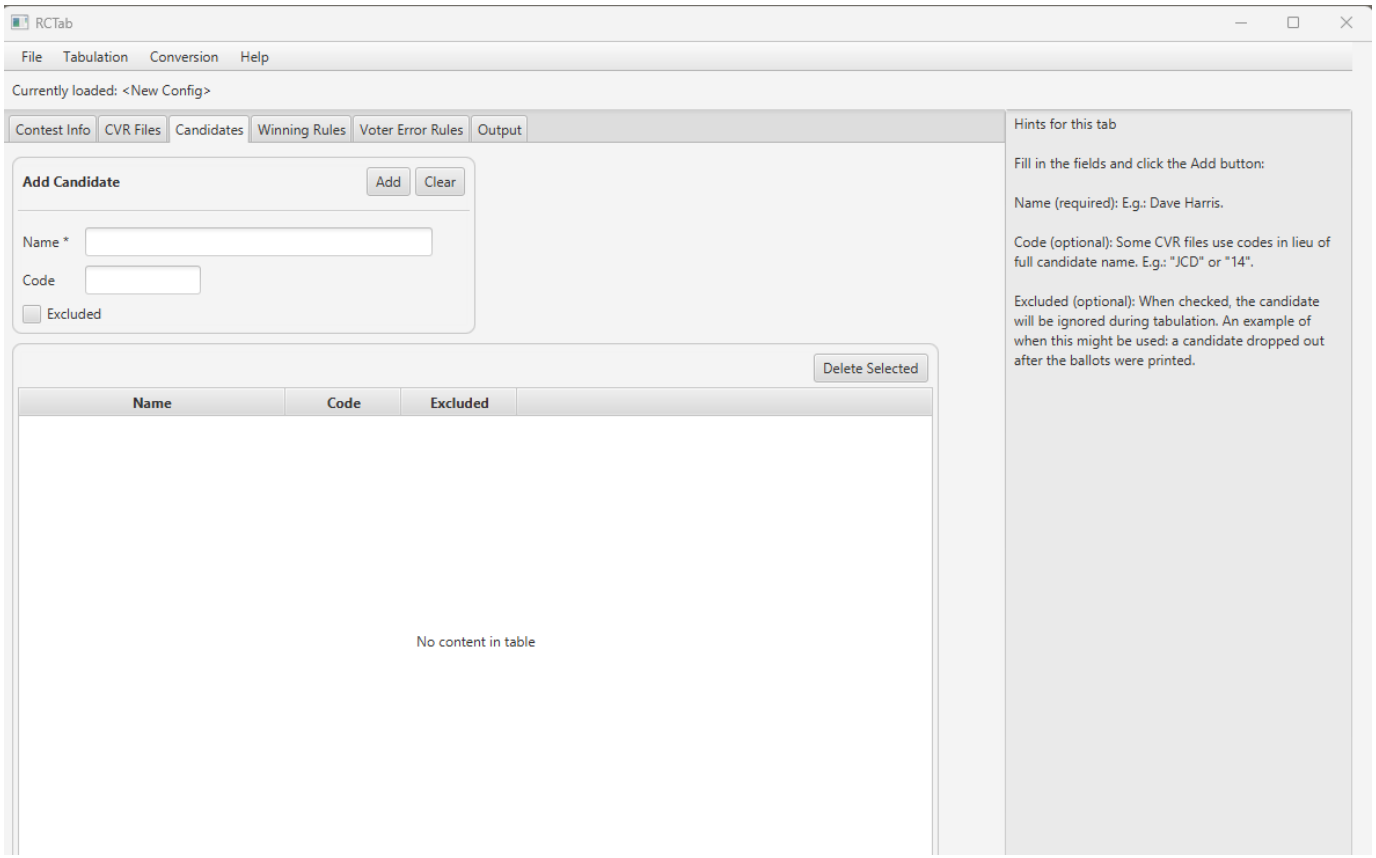


Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

Path: Select the path of the folder that contains signed Hart CVRs.

Candidates Tab

The candidates tab allows users to put in information about how candidates are referred to in cast-vote record files. These settings impact how CVR files are read. Candidate names entered on this tab will also be used to display candidate names in results files. Below is a screenshot of the Candidates Tab when a user first navigates to it.



Fill in the fields and click the Add button:

Name (required)**:** E.g.: Dave Harris. This information is used to display candidate names in results files.

Code (optional)**:** Some CVR files use codes in lieu of the full candidate name. e.g.: "JCD" or "14".

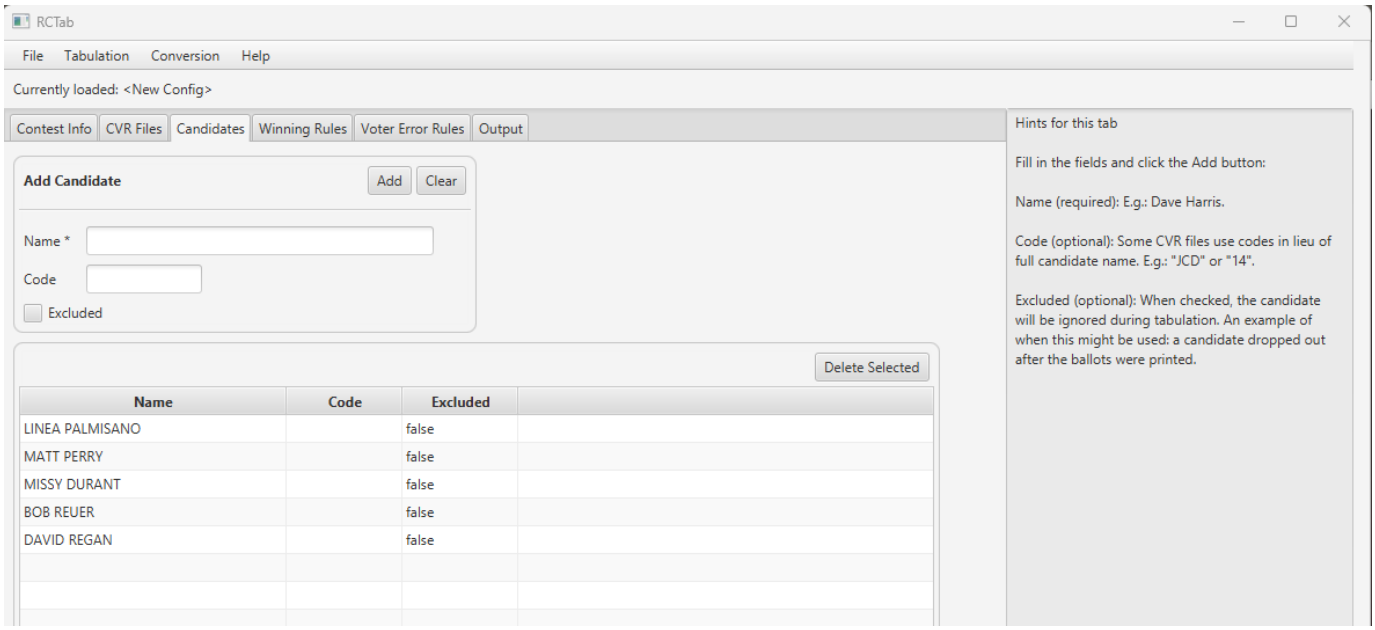
Excluded (optional)**:** When checked, the candidate will be ignored during tabulation. An example of when this might be used: a candidate dropped out after the ballots were printed.

Add: Adds Name, code, and/or excluded information to the Candidates Table.

Clear: Clears any information in Name, Code, and Excluded.

Delete Selected: Deletes selected candidate data from the Candidate Table.

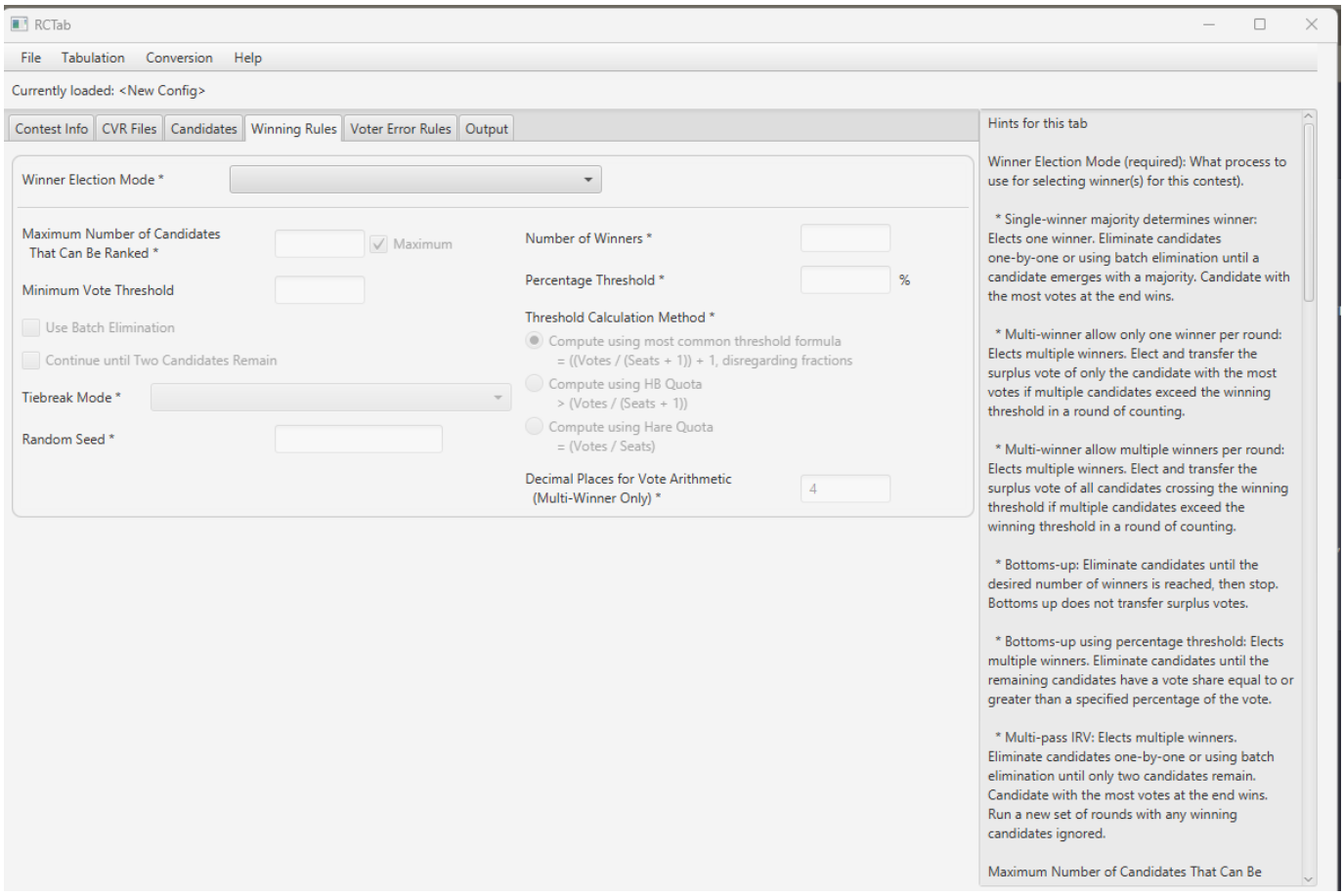
Below is a screenshot of an example of a Candidates Tab while being filled out:



Winning Rules Options

Winning Rules options tell RCTab what kind of ranked choice voting election to run and how to handle details required for each kind of ranked choice voting RCTab can run.

This is a screenshot of what winning rules look like when a user first navigates to them.



There are two main sets of options on the Winning Rules tab: Winner Election Mode options and Tiebreak Mode options.

Winner election mode options have many options that interact in many different ways. Which options are required and which options users can edit ultimately depend on the winner election mode option itself. Options are described below. Screenshots are then provided of each interface for each winner election mode.

Winner Election Mode (required)**:** What process to use for selecting winner(s) for this contest.

- **Single-winner majority determines winner:** Elects one winner. Eliminate candidates one-by-one or using batch elimination until a candidate emerges with a majority. Candidate with the most votes at the end wins.
- **Multi-winner allows only one winner per round:** Elects multiple winners. Elect and transfer the surplus vote of only the candidate with the most votes if multiple candidates exceed the winning threshold in a round of counting.
- **Multi-winner allows multiple winners per round:** Elects multiple winners. Elect and transfer the surplus vote of all candidates crossing the winning threshold if multiple candidates exceed the winning threshold in a round of counting.
- **Bottoms-up:** Eliminate candidates until the desired number of winners is reached, then stop. Bottoms up does not transfer surplus votes.
- **Bottoms-up using percentage threshold:** Elects multiple winners. Eliminate candidates until the remaining candidates have a vote share equal to or greater than a specified percentage of the vote.
- **Multi-pass IRV:** Elects multiple winners. Eliminate candidates one-by-one or using batch elimination until only two candidates remain. Candidate with the most votes at the end wins. Run a new set of rounds with any winning candidates ignored.

Maximum Number of Candidates That Can Be Ranked (required)**:** How many rankings each voter has in this contest. This tells RCTab how many rankings to read from each CVR. This option can be filled out in the text box or by selecting a checkbox. Selecting the checkbox disables the text box option and inserts the value “maximum”. The checkbox can be unselectedDefault value: Maximum. Available in all winner election modes.

Minimum Vote Threshold (optional)**:** The number of first-choice votes a candidate must receive in order to remain in the race. Any candidates falling below the minimum vote threshold are eliminated and have their votes transferred. Most jurisdictions do not set a minimum vote threshold. Note that if no candidate exceeds the minimum vote threshold, vote tabulation will silently fail. Be sure not to set a minimum vote threshold beyond the number of votes candidates have. This option can impact how any winner election mode tabulation runs. Its validity is not impacted by any other option. Available in all winner election modes.

Use Batch Elimination (optional)**:** Batch elimination, or simultaneous elimination of all candidates for whom it is mathematically impossible to be elected, eliminates all candidates who cannot receive enough votes to surpass the candidate with the next highest number of votes. Example: in a six candidate contest with 200 votes, Candidate A has 80 votes, Candidate B has 70, and the other four combined have 50. Because those four candidates can never combine their votes to surpass Candidate B, they can be batch eliminated. This option impacts how single-winner majority determines winner tabulation runs. Its validity is impacted by the winner election mode. It will not impact who wins the election. Available only when Winner Election Mode is "Single-winner majority determines winner".

Continue until Two Candidates Remain (optional)**:** Single-winner ranked choice voting elections can stop as soon as a candidate receives a majority of votes, even though 3 or more candidates may still be in the race. Selecting this option will run the round-by-round count until only two candidates remain, regardless of when a candidate wins a majority of votes. It will not impact who wins the election. Available only when Winner Election Mode is "Single-winner majority determines winner".

Number of Winners (required)**:** The number of seats to be filled in the contest. This option impacts the calculation of the election threshold. Its validity and editability is impacted by the winner election mode. It sets the value for “S” in the calculations described in the UI/configuration file parameters. Available and required for Multi-winner allow only one winner per round, Multi-winner allow multiple winners per round, Bottoms-up, and Multi-pass IRV. Set to 1 by RCTab for Single-winner majority determines winner.

- Note that multi-pass IRV does not use this value to calculate its election threshold. The election threshold in multi-pass IRV is always calculated using an S value of 1. This number is used in multi-pass IRV to determine how many instances of a single-winner majority determines winner tabulation must be run on CVR files. See Configuration Parameters for more details.

Percentage Threshold: The share of votes a candidate must have in order to win. Used to calculate the election threshold in “Bottoms-up using percentage threshold.” Candidates falling below this threshold are eliminated one-by-one beginning with the candidate with the fewest votes. Required and Available only when Winner Election Mode is "Bottoms-up using percentage threshold".

Threshold Calculation Method: The threshold of election is the number of votes a candidate must receive in order to win the election. There are three primary ways to calculate the threshold of election in multi-winner RCV contests. This will be set in law (either by statute or regulation) in your jurisdiction. Required and available only when Winner Election Mode is "Multi-winner allow only one winner per round" or "Multi-winner allow multiple winners per round".

- **Compute using most common threshold formula:** The most common threshold formula is calculated by dividing the number of votes by the number of seats plus one, then adding one to that number. Fractions are disregarded. This is also known as the Droop quota. Candidates must receive this number of votes (or more) to win. This is the default threshold calculation.
- **Compute using HB Quota**: The HB, or Hagenbach-Bischoff, Quota divides the number of votes by the number of seats plus one, leaving fractions. Candidates must receive more than this number of votes to win.
- **Compute using Hare Quota**: The Hare quota divides the number of votes by the number of seats. Fractions are disregarded. It requires candidates to receive that number of votes (or more) to win.

Decimal Places for Vote Arithmetic (Multi-Winner Only): Sets how many decimal places after the decimal point are used in surplus transfers and in calculating the threshold. Its validity and editability is impacted by the winner election mode. Required and available only when Winner Election Mode is "Multi-winner allow only one winner per round" or "Multi-winner allow multiple winners per round". Default value: 4.

TIEBREAKING

There are six tiebreak modes (described in technical detail in **Section 25 - Configuration File Parameters**). Tiebreak modes impact the validity/operation of one other option: random seed. Tiebreak mode is required for all winner election modes.

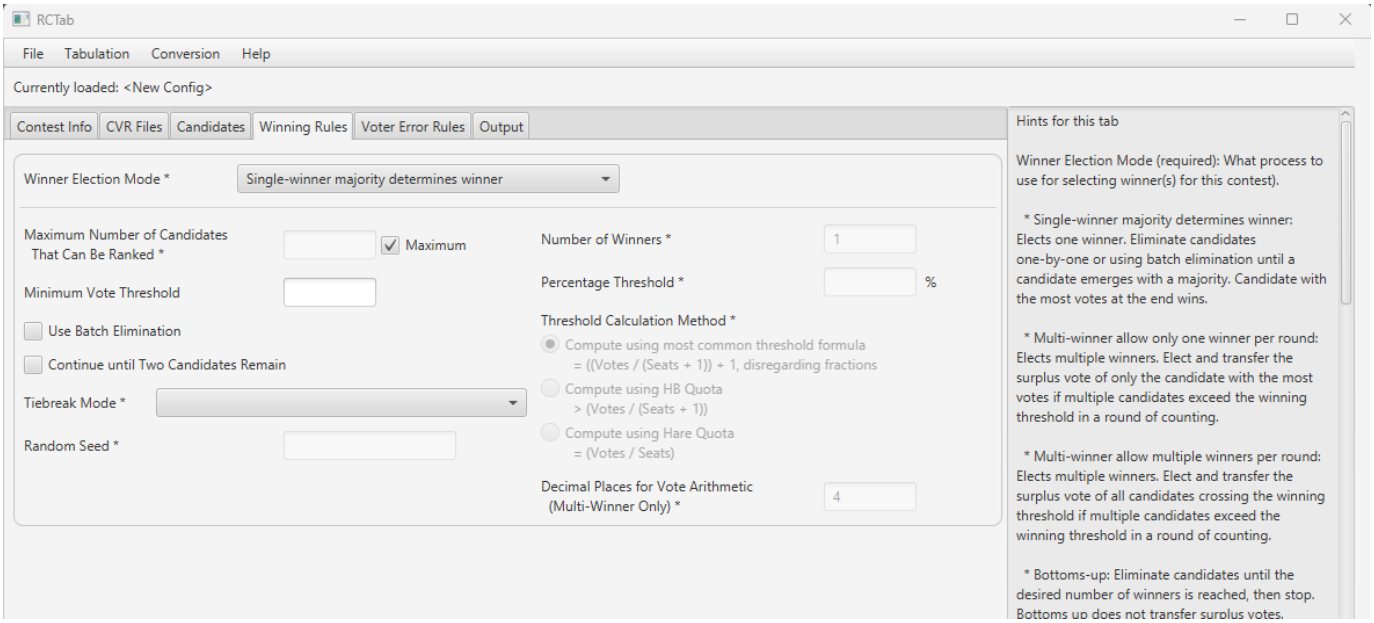
Tiebreak Mode (required for all winner election modes)**:** Ties in ranked choice voting contests can occur when eliminating candidates or when electing candidates. All winner election modes may have a tie for last place. Tiebreak mode breaks the tie and determines which candidate loses the tie. Multi-winner allows only one winner per round contests can have ties between candidates who have both crossed the threshold of election; in that case, ties are broken to determine whose surplus vote value transfers first. Tiebreak procedures are set in law, either in the ranked choice voting law used in your jurisdiction or in the elections code more generally. Select the option from this list that complies with the law and procedure in your jurisdiction.

- **Random:** Randomly select a tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect. Requires a random seed.
- **Stop counting and ask:** Pause count when a tie is reached. The user is prompted to select any tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect.
- **Previous round counts (then random):** The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, the tie is broken randomly. Requires a random seed.
- **Previous round counts (then stop counting and ask):** The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, the user is prompted to select any tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect.
- **Use candidate order in the config file:** Use the order of candidates in the config file to determine tiebreak results. Candidates lower in the list lose the tiebreaker.
- **Generate permutation:** Generate a randomly ordered list of candidates in the contest. Candidates lower in the permutation lose the tiebreaker. Requires a random seed.

Random Seed (required if Tiebreak Mode is "Random", "Previous round counts (then random)", or "Generate permutation"): Enter a positive or negative integer to generate random orders. Impacts operation of randomized functions for tiebreaking. Unavailable if Tiebreak Mode is Stop Counting and Ask, Previous round counts (then stop counting and ask) or Use candidate order in the config file.

Following are screenshots of each of the winning rules interfaces on RCTab.

SINGLE-WINNER MAJORITY DETERMINES WINNER



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Use Batch Elimination

Continue until Two Candidates Remain

Tiebreak Mode*

Random Seed (* if using randomized)

Automatically fills in and locks this option:

Number of Winners*

Invalid options, locked for this mode:

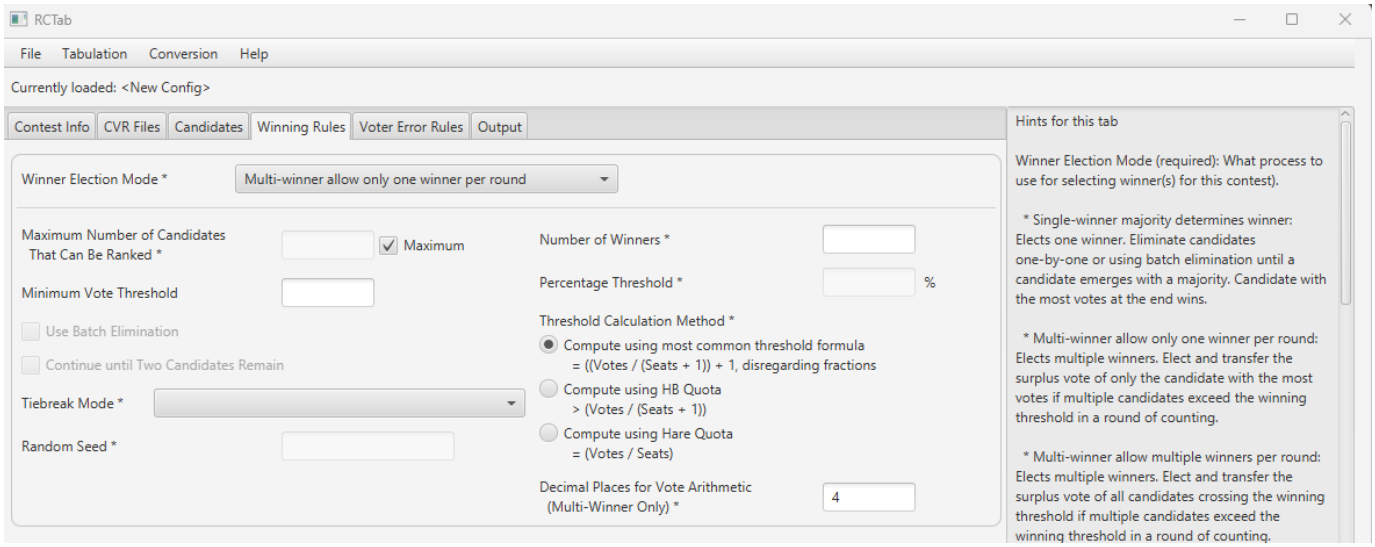
Percentage Threshold

Threshold Calculation Method

Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

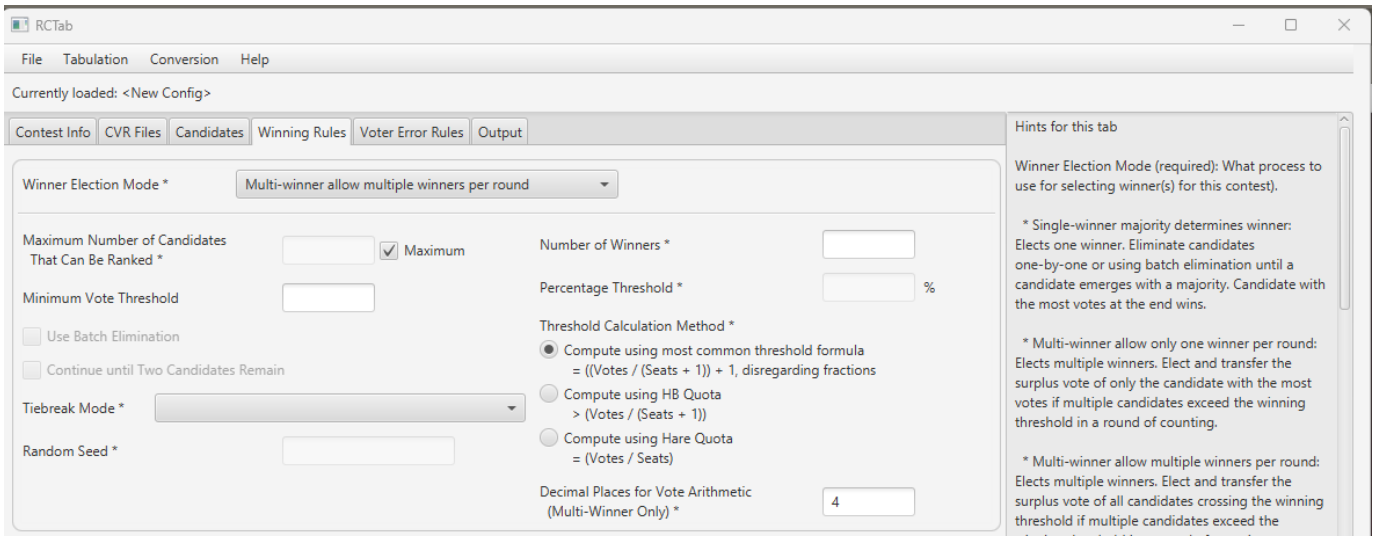
MULTI-WINNER ALLOW ONLY ONE WINNER PER ROUND



Permits user to edit these options:	**Invalid options, locked for this mode:**
Maximum Number of Candidates That Can Be Ranked*	Percentage Threshold
Minimum Vote Threshold	Use Batch Elimination
Tiebreak Mode*	Continue until Two Candidates Remain
Random Seed (* if using randomized tiebreak mode)	
Number of Winners*	
Threshold Calculation Method*	
Decimal Places for Vote Arithmetic*	

Note: Required settings are denoted with an asterisk (*).

MULTI-WINNER ALLOW MULTIPLE WINNERS PER ROUND



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Tiebreak Mode*

Random Seed (* if using randomized tiebreak mode)

Number of Winners*

Threshold Calculation Method*

Decimal Places for Vote Arithmetic*

Invalid options, locked for this mode:

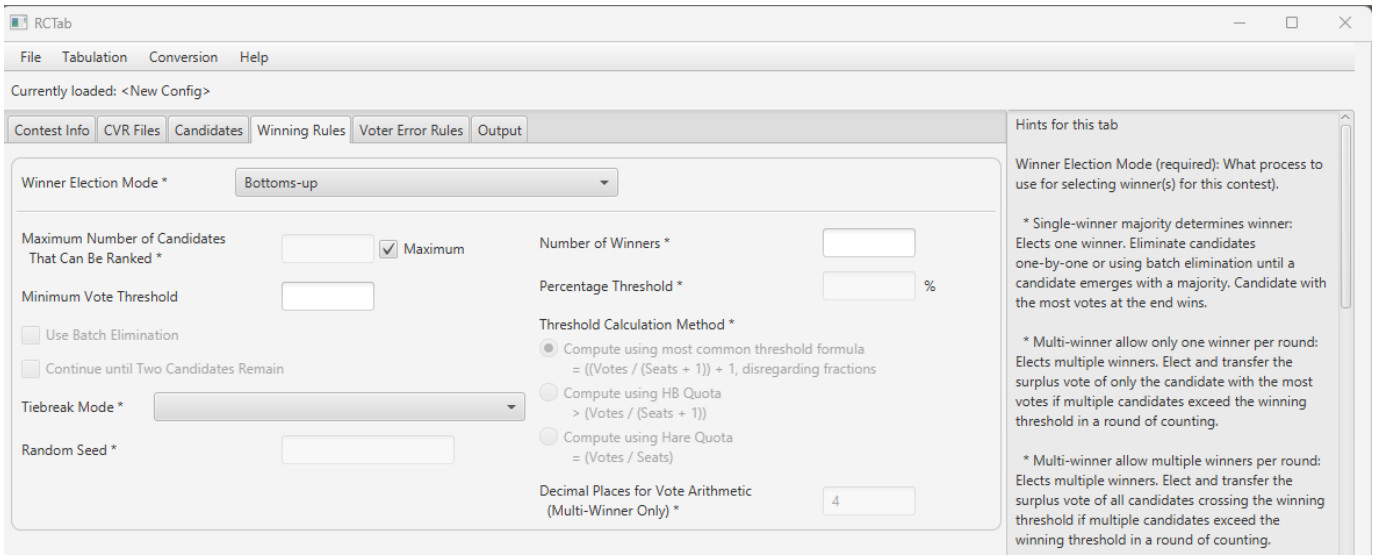
Use Batch Elimination

Continue until Two Candidates Remain

Percentage Threshold

Note: Required settings are denoted with an asterisk (*).

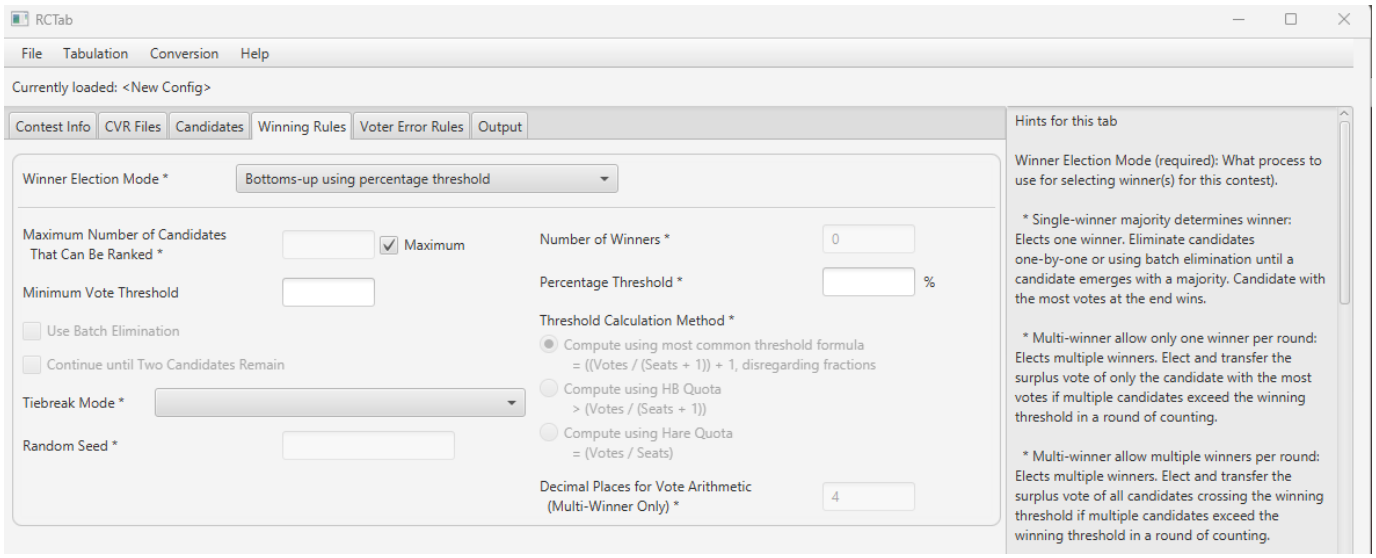
BOTTOMS-UP



Permits user to edit these options:	**Invalid options, locked for this mode:**
Maximum Number of Candidates That Can Be Ranked*	Use Batch Elimination
Minimum Vote Threshold	Continue until Two Candidates Remain
Tiebreak Mode*	Percentage Threshold
Random Seed (* if using randomized tiebreak mode)	Threshold Calculation Method
Number of Winners*	Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

BOTTOMS-UP USING PERCENTAGE THRESHOLD



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Tiebreak Mode*

Random Seed (* if using randomized tiebreak mode)

Percentage Threshold*

Automatically fills in and locks this option:

Number of Winners*

Invalid options, locked for this mode:

Use Batch Elimination

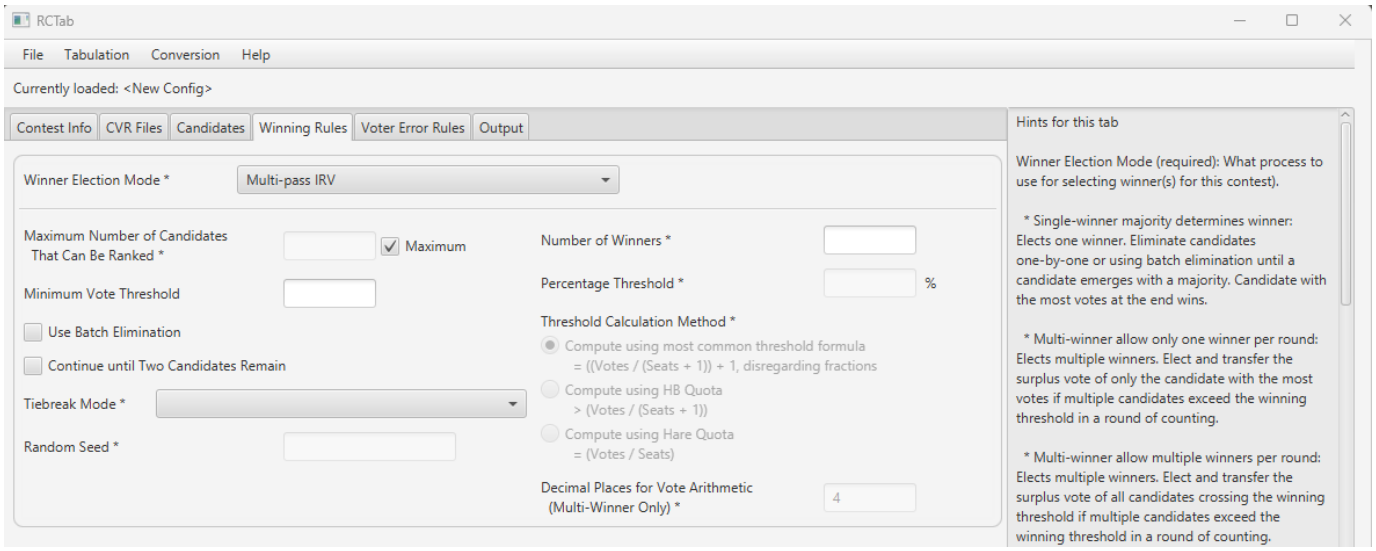
Continue until Two Candidates Remain

Threshold Calculation Method

Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

MULTI-PASS IRV

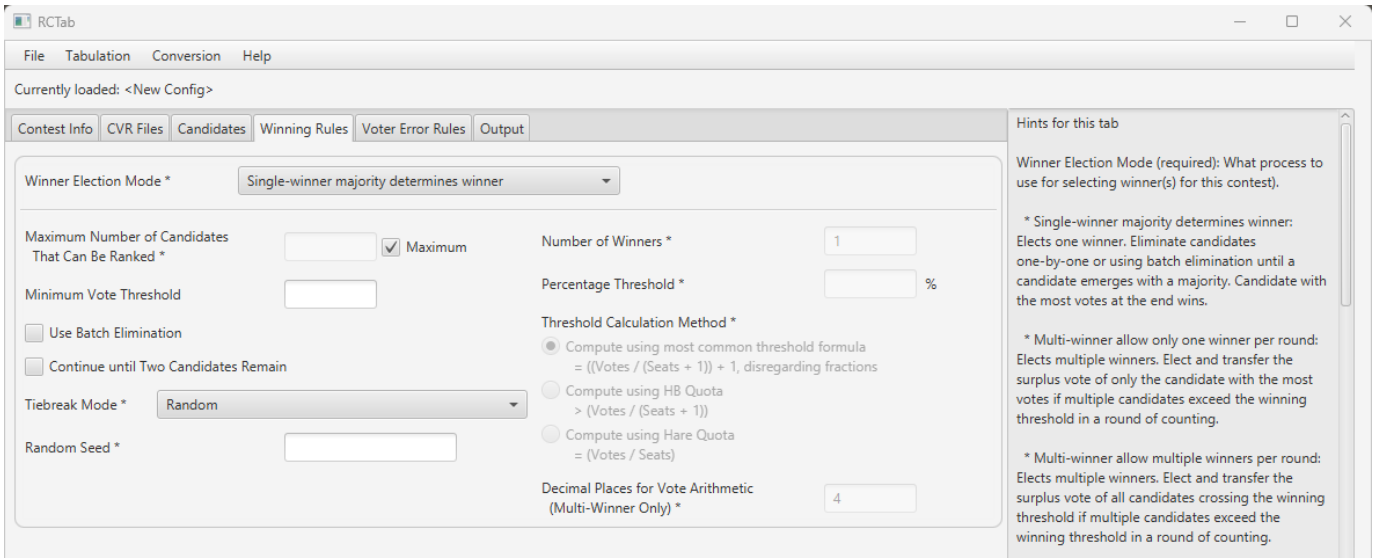


Permits user to edit these options (required settings denoted with *):	**Invalid options, locked for this mode:**
Maximum Number of Candidates That Can Be Ranked*	Use Batch Elimination
Minimum Vote Threshold	Continue until Two Candidates Remain
Tiebreak Mode*	Threshold Calculation Method
Random Seed (* if using randomized tiebreak mode)	Decimal Places for Vote Arithmetic
Number of Winners*	Percentage Threshold

TIEBREAK MODES

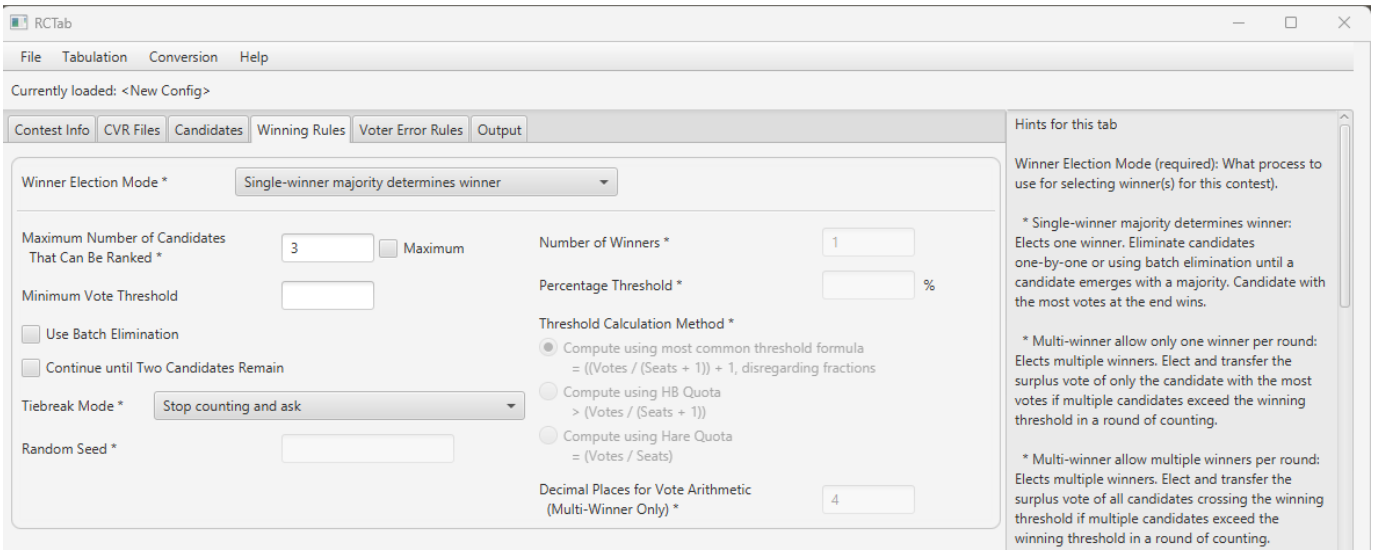
Random, Generate Permutation, Previous Round Counts (then random)

If users select any of the tiebreaking modes that incorporate randomness (random, generate permutation, previous round counts (then random)) this is how RCTab will appear. Users must fill out the Random Seed setting.



Stop counting and ask, previous round counts (then stop counting and ask), use candidate order in the config file

If users select any of the tiebreaking modes that incorporate user input (Stop counting and ask, previous round counts (then stop counting and ask), use candidate order in the config file) this is how RCTab will appear. The Random Seed field will be locked.



Voter Error Rules

The tabulator needs to know how to handle voter errors in your jurisdiction. These requirements are typically included in statute or regulation. This is a screenshot of what winning rules looks like when a user first navigates to it.

Overvote Rule (required)**:** How to handle a ballot where a voter has marked multiple candidates at the same ranking when that ballot is encountered in the round-by-round count.

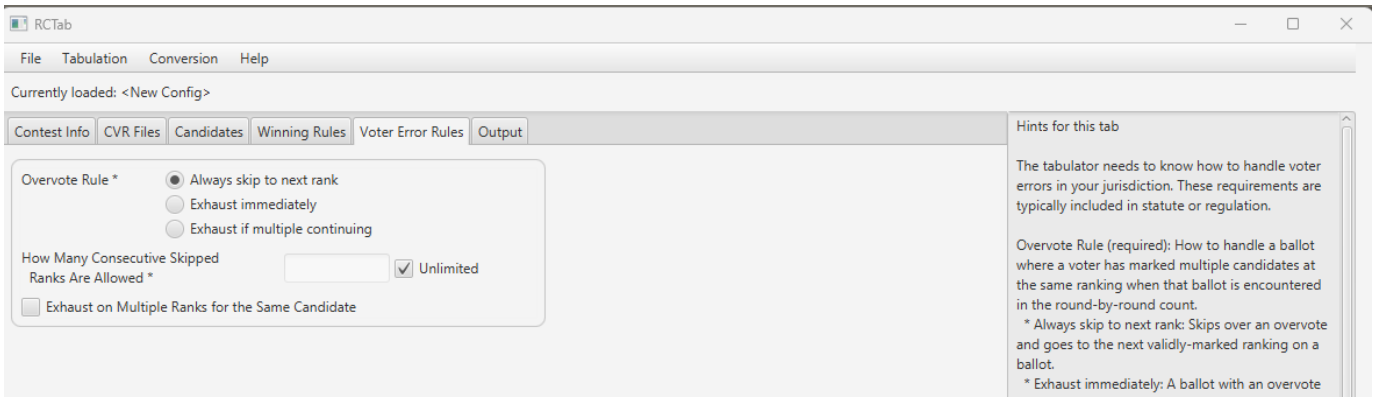
- **Always skip to next rank:** Skips over an overvote and goes to the next validly-marked ranking on a ballot.
- Requires Overvote Label to be supplied (when overvote label can be edited)
- **Exhaust immediately:** A ballot with an overvote exhausts when that overvote is encountered in the rounds of counting.
- Requires overvote label to be supplied (when overvote label can be edited)
- **Exhaust if multiple continuing:** If a voter has an overvote but only one candidate at that overvote is still in the race when that overvote is encountered, the ballot counts for that candidate. If multiple candidates at the overvote are still in the race, the ballot exhausts.
- Requires overvote delimiter to be supplied (when overvote delimiter can be edited)

How Many Consecutive Skipped Ranks Are Allowed (required)**:** How many rankings in a row can a voter skip and still have later rankings count? 0 allows no skipped rankings. `1` allows voters to skip rankings one at a time, but not more than `1` in a row, and so on. Selecting the checkbox disables the text box and inserts the value “unlimited”. Checkbox can be unselected. Default value: `1`. Example: A voter could rank their 1st, 3rd and 5th choices and not exhaust the ballot under this rule, for example.

Exhaust on Multiple Ranks for the Same Candidate (optional)**:** When checked, the tabulator will exhaust a ballot that includes multiple rankings for the same candidate when that repeat ranking is reached. When unchecked repeated rankings will not exhaust the ballot. Example: A voter ranks the same candidate 1st and 3rd, a different candidate 2nd, and another candidate

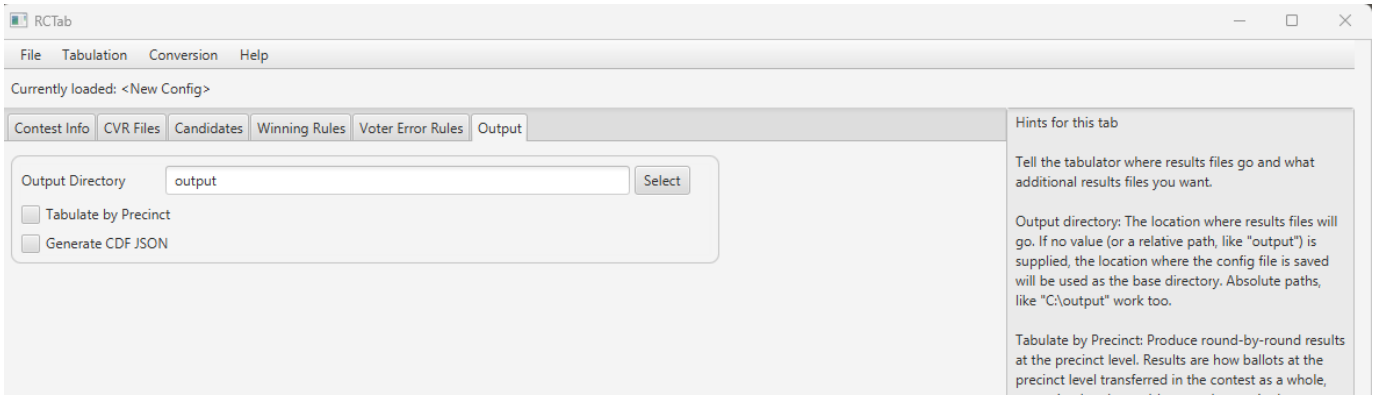
4th. If their original first choice and their second choice are eliminated, the ballot exhausts when it reaches the repeat ranking in rank 3. The ranking in the 4th rank does not count.

Below is an example screenshot of a sample filled-out voter error rules tab. The user has selected Always skip to next rank for Overvote, has selected unlimited for How Many Consecutive Skipped Ranks Are Allowed (making the text box inoperable) and has left Exhaust on Multiple Ranks for the Same Candidate blank.



Output Tab

Tell the tabulator where results files go and what additional results files you want. This is a screenshot of what output looks like when a user first navigates to it.



Output directory: One folder per contest tabulated should be created. Output directory instructs RCTab where to save any output files from successful RCTab tabulations. The location where results files will go. If no value (or a relative path, like `output`) is supplied, the location where the config file is saved will be used as the base directory. Absolute paths, like `C:\output` work too. Select allows users to navigate through Explorer to select a location.

RCTab will not allow the output path to be configured to a Windows user account folder (by default anything under the path `c:\users\`). This requirement ensures read-only output. You **must** follow the instructions in **Section 22 - Installation Instructions for Windows OS** to set permissions on the output folder. Following these instructions ensures that the 'RCTab' user account cannot edit or delete RCTab summary output files or audit logs.

Tabulate by Precinct: Produce round-by-round results at the precinct level. Results are how ballots at the precinct level transferred in the contest as a whole, not a simulated round-by-round count in the precinct. Requires precinct information in CVR Files tab for ES&S CVRs. Impacts which files are written to the output directory.

Generate a CDF JSON: Produce a VVSG common data format JSON file of the CVR. Impacts which files are written to the output directory.

2.23.4 Generating Results Files

Validating Configuration files

Once a configuration file is successfully created, the user must validate the configuration file.

1. Click on "Tabulation" at the top of the software window
2. Click on "Validate"
3. 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.
4. If any error messages appear in the log box, refer to **Section 29 - RCTab Operator Log Messages** and messages in the log box for how to resolve errors. If the error persists, restart the RCTab software

Saving Configuration Files

Once ready to run a tabulation, the user must first save the configuration file.

1. Click on "File" at the top of the software window
2. Click on "Save..."
3. Select a location to save the configuration file. The manufacturer suggests users save the configuration file to the same location set in the Output Directory setting.
4. 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.
5. If any error messages appear in the log box, refer to 430 RCTab Operator Log Messages documentation and messages in the log box for how to resolve errors. If the error persists, restart RCTab software.

Running a Tabulation

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

1. Click on "Tabulation" at the top of the software window
2. Click on "Tabulate"
3. Tabulation will begin.
4. If all the above steps were successfully completed, Tabulation will run until complete.
5. Tabulator log box will update with messages as Tabulation proceeds.
6. Once complete, the Tabulator log box will display a message stating `Results written to: [filepath from Output Directory]`

Output files will be:

- `.csv` contest summary files
- `summary.csv` Whole-contest summary file
- `summary.csv.hash` Corresponding hash file. This contains the hash to verify the results in `summary.csv`
- Precinct-by-precinct summary files (if tabulating by precinct)
- `.json` contest summary files
- `summary.json` Whole-contest summary file
- `summary.json.hash` Corresponding hash file. This contains the hash to verify the results in `summary.json`
- Precinct-by-precinct summary files (if tabulating by precinct)
- `.log` audit files
- `.log` audit files are exported in 50MB sections. If a `.log` file exceeds 50MB an additional `.log` file is started by RCTab
- Corresponding `audit_N.log.hash` file. This contains the hash to verify the information in each `audit_N.log file`
- `.json` CDF (common data format) files if Generate a CDF JSON is checked

If necessary, instructions for verifying result file hashes can be found in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**.

Users can then navigate to "File" and click "Exit" if all contests are tabulated.

If more contests remain to be tabulated, and contests will contain fewer than 1,000,000 total votes, user can navigate to “File” and click “New.” This will clear all fields in RCTab and permit the user to create a new configuration file.

If contests to be tabulated will contain more than 1,000,000 total votes, return to start of guide and re-launch tabulator according to large configuration launch requirements. Then follow the guide to set up a configuration file.

If any errors arise in the use of RCTab, refer to **Section 29 - RCTab Operator Log Messages**. Errors arising out of any hardware or software other than RCTab should refer to **Section 09 - System Maintenance Manual** and any relevant user and maintenance manuals.

Before publishing results, jurisdictions should use their established reconciliation procedures to ensure total votes counted in each round equals total ballots cast in the contest. If numbers in the reconciliation process do not match, the user should double-check that all CVRs for that contest were exported successfully from the voting system and run RCTab process for that contest again. Rely on user jurisdiction CVR handling procedures for transmitting CVRs.

Any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team of no less than two trained personnel.

Required capabilities that may be bypassed or deactivated during installation or operation by the user shall be clearly indicated. Additional capabilities that function only when activated during installation or operation by the user shall be clearly indicated. Additional capabilities that normally are active but may be bypassed or deactivated during installation or operation by the user shall be clearly indicated.

The installation process for RCTab software does not give users the opportunity to bypass or deactivate options or settings.

Capabilities that are active or inactive in software operation depend on various factors. Many of these factors are laid out above. Configuration files determine which system capabilities apply to a given set of voting data. More information about operations that users can set through the user interface is provided in **Section 25 - Configuration File Parameters**. Information about the operation of those settings is also provided in **Section 02 - Software Design and Specifications.**

2.23.5 Configuration File Parameters and UI Label Match Sheet

Below is a list of all configuration file parameter labels and their corresponding labels in RCTab UI. Labels are organized by the order of their appearance in the RCTab UI.

Configuration File Parameters Name/Label	UI Name/Label
`contestName`	Contest Name
`contestDate`	Contest Date
`contestJurisdiction`	Contest Jurisdiction
`contestOffice`	Contest Office
`rulesDescription`	Rules Description
`provider`	Provider
`cdf`	CDF
`clearBallot`	Clear Ballot
`Dominion`	Dominion
`ess`	ES&S
`hart`	Hart
`filePath`	Path
`contestId`	Contest ID
`firstVoteColumnIndex`	First Vote Column
`firstVoteRowIndex`	First Vote Row
`idColumnIndex`	ID Column
`precinctColumnIndex`	Precinct Column
`overvoteDelimiter`	Overvote Delimiter
`overvoteLabel`	Overvote Label
`undervoteLabel`	Undervote Label
`undeclaredWriteInLabel`	Undeclared Write-in Label
`treatBlankAsUndeclaredWriteIn`	Treat Blank as Undeclared Write-In
`name`	Name
`code`	Code
`excluded`	Excluded
`winnerElectionMode`	Winner Election Mode
`singleWinnerMajority`	Single-Winner Majority Determines Winner
`multiWinnerAllowOnlyOneWinnerPerRound`	Multi-Winner Allow Only One Winner Per Round
`multiWinnerAllowMultipleWinnersPerRound`	Multi-Winner Allow Multiple Winners Per Round
`bottomsUp`	Bottoms-up
`bottomsUpUsingPercentageThreshold`	Bottoms-up using Percentage Threshold
`multiPassIrv`	Multi-Pass IRV
`maxRankingsAllowed`	Maximum Number of Candidates That Can Be Ranked
`minimumVoteThreshold`	Minimum Vote Threshold

Configuration File Parameters Name/Label	UI Name/Label
`batchElimination`	Use Batch Elimination
`continueUntilTwoCandidatesRemain`	Continue Until Two Candidates Remain
`tiebreakMode`	Tiebreak Mode
`random`	Random
`stopCountingAndAsk`	Stop counting and ask
`previousRoundCountsThenRandom`	Previous Round Counts (then random)
`prevousRoundCountsThenAsk`	Previous Round Counts (then stop counting and ask)
`useCandidateOrder`	Use candidate order in config file
`generatePermutation`	Generate permutation
`randomSeed`	Random Seed
`numberOfWinners`	Number of Winners
`multiSeatBottomsUpPercentageThreshold`	Percentage Threshold
`nonIntegerWinningThreshold`	Compute using HB Quota > (Votes / (Seats + 1))
`hareQuota`	Compute using Hare Quota = (Votes/Seats)
`decimalPlacesForVoteArithmetic`	Decimal Places for Vote Arithmetic (Multi-Winner Only)
`overvoteRule`	Overvote Rule
`alwaysSkipToNextRank`	Always skip to next rank
`ExhaustImmediately`	Exhaust Immediately
`exhaustIfMultipleContinuing`	Exhaust if multiple continuing
`maxSkippedRanksAllowed`	How Many Consecutive Skipped Ranks Are Allowed
`exhaustOnDuplicateCandidate`	Exhaust on Multiple Ranks for the Same Candidate
`outputDirectory`	Output Directory
`tabulateByPrecinct`	Tabulate by Precinct
`generateCdfJson`	Generate CDF JSON

2.24 Section 19 - Tabulation Options for RCV Tabulation

Ranked choice voting is an umbrella term for a collection of voting methods where voters rank candidates in order of preference and votes are counted in rounds. Ranked choice voting elections occur in a series of rounds. Those rounds can consist of counting votes, eliminating candidates, transferring votes from eliminated candidates, electing candidates, and transferring surplus votes from elected candidates. Ranked choice voting is counted at both the granular ballot-by-ballot level and at a contest-wide level. In other words, counting RCV requires knowing details of how every individual ballot is marked as well as the total number of votes for each candidate at each stage of the RCV count.

The following is an enumeration and discussion of the various tabulation options that exist for Ranked Choice Voting (RCV) elections. Also included is a glossary of ranked choice voting terms. This same glossary is included in **Section 18 - User Guide**.

Counting RCV elections proceeds as follows: all first-choice rankings are counted first. If no candidate receives a specified number of votes necessary for election (typically known as the threshold of election), then the candidate with the fewest votes is eliminated. Ballots counting for that candidate then transfer to the next-ranked candidate on each ballot. These rounds of counting, elimination, and redistribution recur until all seats up for election are filled. In some multi-winner variations of RCV, candidates who receive more votes than necessary to be elected – those candidates whose vote totals exceed the threshold of election – are deemed to have surplus votes. If a candidate has a surplus, the next step in counting is to transfer that surplus. Surplus votes are transferred in various ways, but they follow the same logic as elimination transfers: each ballot counting for a candidate with surplus votes transfers vote value to the next-ranked candidate on that ballot.

There are many variations used in processing contest selections to determine the candidates that are elected. Voting the ballot is the same amongst the variants although the presentation of ballot design may vary. This document addresses all variations currently in use in the United States, which fall into two major categories: single-winner RCV and variants of RCV electing multiple candidates. **single-winner RCV** is also known as instant runoff voting (IRV). Variants of RCV that elect multiple candidates include **multi-winner ranked choice voting** (also known as the single transferable vote (STV) and proportional ranked choice voting), **multi-pass IRV** (also known as block-preferential voting), **bottoms-up RCV**, and **bottoms-up RCV with percentage threshold**. All use multi-round processing methods to determine the outcome(s) and require the full set of ballots or cast vote records (CVRs), for RCV contests to be processed. Purposes of this method include avoiding holding a separate runoff election, eliminating the effect of vote splitting or implementing proportional representation in multi-member representative bodies.

The processing used in each single-winner RCV round, if there is no apparent winner, is as follows. The candidate with the lowest vote total in the round is considered eliminated. Each CVR containing the eliminated candidate as the current 1st choice is processed to count for the next highest-ranked continuing candidate (one that has not been eliminated). If there are no choices left on a given ballot, that ballot is considered exhausted and inactive for any subsequent rounds. The new round tabulates totals and determines whether a candidate now has sufficient votes to be elected. If not, the process is repeated round-by-round until a candidate has sufficient votes to win.

In single-winner RCV, once a candidate has sufficient votes to be elected (This typically means once a candidate has a majority of votes in a round, but see item 1 for more detail.) tabulation will stop.

In bottoms-up RCV, bottoms-up RCV with percentage threshold, and multi-pass IRV, tabulation proceeds in much the same way. Votes are counted, candidates with the fewest votes are eliminated, and votes transfer to the next candidate on each of those ballots. In bottoms-up RCV, these rounds of counting recur until there are as many candidates left as there are seats to fill. In bottoms-up RCV with percentage threshold, these rounds of counting recur until all candidates remaining have a share of votes equal to or greater than the percentage threshold. In multi-pass IRV, counting proceeds slightly differently. Multi-pass IRV uses single-winner IRV for multi-winner contests by electing candidates one at a time. The same process for counting rounds as described above is followed. The count is then reset and run again, but any rankings for an elected candidate are ignored. These passes of IRV are run multiple times until all seats are filled. This method is not designed to guarantee proportional representation, in contrast to STV which was designed with the intent to provide proportional representation.

In multi-winner ranked choice voting (aka single-transferable vote or proportional RCV), rounds operate somewhat differently. Elimination operates in the same way as all of the above: If no candidate receives a previously specified number of votes necessary for election (typically known as the threshold of election), then the candidate with the fewest votes is eliminated. Ballots counting for that candidate then transfer to the next-ranked candidate on each ballot. Where counting differs, however, is when a candidate receives enough votes to win. If a candidate receives more votes than necessary to be elected – if their vote

total exceeds the threshold of election – they are deemed to have surplus votes. If a candidate has a surplus, the next step in counting is to transfer that surplus. Surplus votes are transferred in various ways, but they follow the same logic as elimination transfers: each ballot counting for a candidate with surplus votes transfers vote value to the next-ranked candidate on that ballot. In multi-winner RCV, rounds of elimination and election recur until all seats up for election are filled.

Note that in both multi-pass IRV and multi-winner RCV, once a candidate is declared elected they cannot receive any more votes from vote transfers.

There is variation in how each of the above tabulation methods actually operate on the round-by-round and ballot-by-ballot level. Variations related to how overvotes, repeated ranking, skipped rankings, candidate elimination, surplus transfers, thresholds, ties and other factors are handled. Each of these possible variations is described above. When a variation applies to a specific type of RCV, that will be noted. Example variations are briefly described in the bulleted list below:

- Handling an overvoted choice during round-by-round processing – Is the ballot considered exhausted or is the choice skipped and the ballot checked for a subsequent continuing candidate?
- Handling an omitted choice/ranking during round-by-round processing – Is the choice skipped? Does this cause the ballot to exhaust, are skipped rankings ignored or is one skipped ranking ignored but two skipped rankings in succession exhaust the ballot?
- Handling of a repeat ranking (selecting the same candidate for more than one ranking) during the round-by-round processing – Is the repeated rank skipped in the same way as an omitted ranking or is the ballot considered exhausted?
- Handling of candidate elimination – Are all candidates who have no mathematical chance of advancing concurrently eliminated in the first round only, in any round or is only one candidate eliminated in any round and multiple eliminations of candidates not used?
- When to terminate tabulation – Is tabulation of voted 1st choices used to determine if a candidate(s) has sufficient votes to be elected (thus avoiding the use of an RCV algorithm) or is the algorithm always used? If used, does calculation of the threshold for election include all ballots cast (include over- and undervote totals) or is it based on total valid selections?

Most tabulation options are laid out explicitly in RCV laws, while some laws cover only some of these options, and some of these options are routinely excluded from RCV laws.

1. Termination of tabulation (single-winner)

a. Declare winner if a candidate receives a majority of 1st choice votes in the initial count. See (d) as well. (used in State of Maine, New York City)

i. If winner emerges from initial count, RCTab will not be used.

b. Declare winner when a candidate receives a majority of valid votes in any round (used in Basalt, Benton County, Berkeley, Las Cruces, Oakland, Payson, Portland, ME., San Leandro, Santa Fe, Takoma Park, MD., Vineyard)

i. Setting in RCTab: Winner Election Mode: Single-winner majority determines winner

c. Declare winner when a candidate receives a majority of all votes cast in contest, or most votes when two candidates remain (used in Minneapolis, St. Louis Park St. Paul, Minnesota jurisdictions)

i. Not included in RCTab

d. Declare a winner when only two candidates remain. (used in Eastpointe, State of Maine, New York City, San Francisco, CA.)

i. Setting in RCTab:

i. Winner Election Mode: Single-winner majority determines winner, AND

ii. Select "Continue until Two Candidates Remain"

While any candidate achieving a majority of votes under (a), (b), or (c) will also be the winner under (d), the latter provides a more complete picture of the strength of support for the winner across all voters. While there is some incentive to choose option (a) if it saves on the cost of accumulating all cast vote records (CVRs) for RCV tabulation, if the CVRs are readily available, fully automated tabulation systems can go from (a) to (d) in seconds. San Francisco law requires (b) but the city opts to continue tabulation in line with (d). As with other options, (c) does not alter who will win an election but it does alter how contest results are presented. It may help to envision these different rules as different formulas. These numbers depend on the number of valid ballots in a round – how many ballots can be counted towards a candidate in the ranked choice voting contest in question.

A majority is calculated in (b) jurisdictions using the following formula:

`T + B / (S+1) +1`, disregarding fractions.

T = Threshold

B = Number of Continuing Ballots in the given round of counting

S = Number of Seats to be Filled

A majority is calculated in (c) jurisdictions using the following formula:

T+B/(S+1)+1, disregarding fractions.

T = Threshold

B = Number of Valid Ballots Cast in the first round of counting

S = Number of Seats to be Filled

2. Determining which candidate(s) to eliminate (varies only in single-winner)

a. Lowest vote-getter (default in all methods)

i. Setting in RCTab: Don’t select "Use Batch Elimination"

b. Batch elimination (available only for single-winner contests)

i. Setting in RCTab: Select "Use Batch Elimination"

There are two ways to eliminate candidates in single-winner RCV elections - one at a time or in batches. Eliminating candidates one at a time is straightforward - determine which candidate has the fewest votes and eliminate that candidate.

Eliminating candidates in batches, using batch elimination, is somewhat more complicated. Eliminating candidates using batch elimination makes it faster to count a ranked choice voting election with many candidates - it is possible to have fewer rounds of counting if multiple candidates can be eliminated at once.

Batch elimination eliminates all candidates who cannot receive enough votes to surpass the candidate with the next highest number of votes.

Example: in a six candidate contest with 200 votes, Candidate A has 80 votes, Candidate B has 70, and the other four combined have 50. Because those four candidates can never combine their votes to surpass Candidate B, they can be batch eliminated.

Rule: eliminate the set of one or more candidates C1, C2, C3….Cn if sum of all votes for C1….Cn < votes for candidate B and votes for B < votes for candidate

If a specified candidate satisfies both of the following conditions, then all candidates with fewer votes may be designated as defeated:

a. At least one other candidate has at least as many votes as the specified candidate.

b. The specified candidate has more votes than the total votes for all candidates with fewer votes.

In practice, when tabulating using RCTab or other computerized methods of counting, batch elimination has no effect on how long it takes to count the election. Batch elimination can reduce the number of rounds of counting required for an election, however. And if hand counting, batch elimination does provide a way to count an election more quickly if candidates can be batch eliminated.

3. Overvotes (two or more candidates marked with the same ranking)

a. Skip to next highest-ranked candidate (used in Minneapolis, St. Louis Park, St. Paul)

i. Setting in RCTab: "Overvote Rule: Always Skip to Next Rank"

b. Exhaust ballot (used in Benton County, Berkeley, Eastpointe, Las Cruces, State of Maine, New York City, Oakland, Payson, Portland, San Francisco, Santa Fe, San Leandro, Telluride, Vineyard)

i. Setting in RCTab: "Overvote Rule: Exhaust Immediately"

c. Exhaust ballot unless only one overvoted candidate is a continuing candidate (used in Takoma Park, MD)

i. Setting in RCTab: "Overvote Rule: Exhaust if multiple continuing"

d. Suspend ballot until only one overvoted candidate is a continuing candidate then cast vote for that candidate or any subsequent highest-ranked continuing candidate. (Not currently used)

i. Not included in RCTab

Options (a), (b), and (c) are in use in current RCV jurisdictions. The argument in favor of exhausting the ballot, option (b), is that this makes no judgements about voter intent. An error is an error and that terminates the ballot. While this is consistent with the treatment of overvotes in conventional plurality elections, RCV offers other valid options that keep a voter's ballot alive.

Skipping to the next highest-ranked candidate, option (a), is also used in a number of jurisdictions. This acknowledges the error but does not make it a terminal error. It simply treats the next highest-ranked choice as the first valid choice, which also avoids imposing any outside judgment about voter intent.

Exhausting the ballot unless only one overvoted candidate is a continuing candidate, option (c), simply acknowledges that no overvote exists if only one of the marked candidates is a continuing candidate. This is because each round of tabulation only looks at votes for continuing candidates and ignores candidates that have been eliminated. There is, however, an inequity in this method in that it has unintended effects depending on the ranking at which the overvote occurs. No candidates are eliminated prior to the first round of tabulation, thus, any overvote at the first choice level would exhaust the ballot where an overvote at a later round might not exhaust the ballot.

Option (d) is suggested as a means of eliminating the inequity of the different treatment of overvotes in different tabulation rounds. This option appears to maximize the opportunity for a voter's ballot to be counted and counted as that voter intended. It should be noted, however, this option has not been and is currently not used in any RCV jurisdiction.

4. Skipped rankings

a. Skip to next highest-ranked candidate (used in Berkeley, Las Cruces, Minneapolis, New York City, Oakland, St. Louis Park, St. Paul, San Francisco, Santa Fe, Telluride)

i. Setting in RCTab: Select "Unlimited" for "How Many Consecutive Skipped Ranks are Allowed"

b. Exhaust ballot (used in Colorado)

i. Setting in RCTab: Set "How Many Consecutive Skipped Ranks are Allowed" to `0`

c. Exhaust ballot if two consecutive skipped rankings are encountered (used in Eastpointe, State of Maine, Takoma Park, MD., CA. Senate Bill 212, 2019)

i. Setting in RCTab: Set "How Many Consecutive Skipped Ranks are Allowed" to `1`

Skipping to the next highest-ranked continuing candidate without a limitation in consecutive skips, option (a), runs the risk of casting a ballot for a voter's last choice candidate if they rank only their first and last choices and the first choice is eliminated.

While exhausting a ballot after consecutive skipped rankings, option (c), minimizes this prospect of option (a), it does not entirely eliminate it (e.g., three candidates competing for one seat).

Exhausting a ballot when a skipped ranking is encountered, option (b), again penalizes the voter in a draconian manner for what may be a simple oversight.

Option (c) appears to minimize the potential adverse effects while maximizing the opportunity for the voter's ballot to be counted as intended.

5. Repeat rankings of the same candidate (sometimes referred to as duplicate ranking)

a. Ignore repeat rankings (used in Minneapolis, St. Paul, State of Maine, New York City, Oakland, San Leandro, Berkeley, Portland, ME., Takoma Park, MD., Santa Fe)

i. Setting in RCTab: Don't select "Exhaust on Multiple Ranks for the Same Candidate"

b. Exhaust ballot at repeat ranking (used in Eastpointe)

i. Setting in RCTab: Select "Exhaust on Multiple Ranks for the Same Candidate"

Most jurisdictions are silent on this question in their RCV legislation. Information in this section is derived from practices followed in most jurisdictions. A standard tabulation algorithm will not detect a repeat ranking unless or until the candidate is eliminated. Once eliminated, the algorithm will simply look for the next highest-ranked continuing candidate and will pass over the repeat ranking as it would any other eliminated candidate.

The practice of exhausting a ballot when a repeat ranking is encountered, option b, appears to emerge from a position that all voter "errors" should invalidate a ballot.

6. Ties among last-place candidates

a. Decide by lot (used in Benton County, Berkeley, Eastpointe, State of Maine, Minneapolis, New York City, Oakland, Payson, Portland, ME., St. Louis Park, St. Paul, San Francisco, San Leandro, Santa Fe, Telluride, Vineyard, CA Senate Bill 212, 2019)

i. Setting in RCTab: Tiebreak Mode: Stop counting and ask

b. Most votes in previous round (used in Cambridge, Takoma Park, MD)

i. Setting in RCTab: Tiebreak Mode: Previous Round Counts then Stop counting and ask

c. Predetermine tiebreaking order and include in configuration or algorithm. (Allowed in State of Maine)

i. Setting in RCTab: Tiebreak Mode: Use candidate order in the config file or Generate config

Last place ties pose a different kind of challenge in RCV elections than encountered in plurality elections. Here, time is of the essence since the continuation of the tabulation is dependent upon the resolution of a tie to determine which candidate is eliminated. Many states require that tied candidates be present for a tie resolution, something that is not desirable in RCV last-place ties where a tie must be broken before the tabulation can proceed.

While "most votes in the previous round" generally is an easy, built-in tie-breaking mechanism, it suffers from one major drawback: it may violate the one-person-one-vote principle. This is due to the fact that it gives heavier weight to votes in the previous round than to votes in the current round in deciding who advances and who is eliminated. While this has not yet been litigated, it opens the door to that prospect.

7. Last round ties/ties between candidates to elect

a. Defer to State law (used in Benton County, Berkeley, Maine, New York City, Oakland, San Francisco, San Leandro, Santa Fe, Telluride)

i. Setting in RCTab: Tiebreak Mode: Stop counting and ask

b. Decide by lot (used in Cambridge, Eastpointe, Minneapolis, St. Louis Park, St. Paul, Payson, Portland, ME., Santa Fe, Vineyard)

i. Setting in RCTab: Tiebreak Mode: Stop counting and ask

c. Most votes in previous round (used Takoma Park, MD)

i. Setting in RCTab: Tiebreak Mode: Previous Round Counts then Stop counting and ask

d. Predetermine tiebreaking order and include in configuration or algorithm. (Allowed in State of Maine)

i. Setting in RCTab: Tiebreak Mode: Use candidate order in the config file or Generate config

Last round ties in RCV elections are very similar to ties in plurality elections. Traditional tie breaking protocols may be employed if desired. In all jurisdictions that defer to State law, except New York City, state law requires the use of lots to decide ties. New York

City law defers to party committees in primary elections and appears to be silent on how to decide ties in other circumstances. New York City has decided to decide these ties by lot.

The option of using "most votes in the previous round" here is likely more problematic than in last-place tie situations in that a tie break for the win is more likely to be subject to litigation.

8. RCV threshold calculation (multi-winner, bottoms-up RCV w/ percentage threshold)

a. Multi-winner thresholds

i. Whole number threshold aka Droop Quota or most common threshold (used in Eastpointe, Minneapolis, Cambridge, MA.)

i. Setting in RCTab: Threshold Calculation Method: Compute using most common threshold formula

ii. Fractional threshold aka HB (Hagenbach Bischoff) Quota (not used in US elections, but detailed in CA. Senate Bill 1288, 2017, CA. Senate Bill 212, 2019)

i. Setting in RCTab: Threshold Calculation Method: Compute using HB Quota

iii. Hare Quota (not used in official United States elections)

i. Setting in RCTab: Threshold Calculation Method: Compute using Hare Quota.

b. Bottoms-up RCV with percentage threshold (used in Alaska, Hawaii, Kansas, and Wyoming) 2020 Democratic Presidential Primaries

i. Percentage of votes cast in a round

i. (for discussion of this scroll to the bottom of this section)

ii. Setting in RCTab: Percentage Threshold

Multi-winner RCV uses a calculation based on the number of seats to be elected to determine what share of the votes cast in an election a candidate needs in order to win an election. That threshold is set by the formula floor(T=B/(S+1)+1) (for whole number thresholds), T>B/(S-1)+n (for fractional thresholds), or floor(T=B/S) (for Hare Quota)

T = Threshold

B = Number of Valid Ballots Cast in election

S = Number of Seats to be Filled

S is set by Number of Winners in RCTab.

n = a fraction greater than 0 and less than 1

In percentages, this means the Droop or HB threshold in a three-seat election is 25%+ , a four-seat election is 20%+, a five-seat election is ~16.67%+, and so on. The Hare threshold/quota in a three-seat election is ~33.3% , a four-seat election is 25%, a five-seat election is 20%, and so on.

While single-winner contests under options 1(a), 1(b), and 1(c) involve whole number thresholds, i.e., one vote more than a majority, multi-winner contests may use either whole number or fractional thresholds. This is due to the use of surplus transfers in multi-winner contests to credit a proportional share of extra votes a candidate receives (those above the threshold) to the next highest-ranked choice on ballots cast for a winner.

Fractional thresholds provide for a higher degree of precision in deciding winners or eliminations. For instance, in a three-seat contest where 300 votes are cast, would the threshold be 101, 100.1, 100.01, 100.001, etc. votes? In most elections, such precision would make no difference, however, there are certainly cases where fractions of a vote could make the difference between winning, losing, or tying. This is particularly true in small elections or where there are numerous candidates, many of whom have weak support and the order of elimination is important.

Some jurisdictions define the threshold as T>B/(S+1) in which case the winner must receive a vote total greater than the threshold rather than greater than or equal to.

Calculating the threshold in bottoms-up RCV with percentage threshold is accomplished by multiplying the number of continuing ballots in a round of counting by the percentage threshold set for that election.

T = B * P

```
T = Threshold
```

B = Number of continuing ballots in a round

P = Percentage threshold for this election.

P is set by Percentage Threshold in RCTab

For example, if the percentage threshold is 15% and 2000 ballots count in a round of counting, the threshold will be 300 votes.

9. Multi-winner RCV surplus transfers

a. Fractional transfers (used in Eastpointe, MI, Minneapolis, CA.SB 1288, 2017, CA. SB 212, 2019)

i. All multi-winner RCTab Winner Election Modes use fractional transfer.

b. Whole ballot transfer (used in Cambridge, MA.)

i. Not included in RCTab

Candidates elected in multi-winner RCV elections frequently receive more votes than they need to win an election. These votes in excess of the threshold are called "surplus votes."

To ensure proportionality in multi-winner RCV elections, those surplus votes are transferred to later-ranked candidates on ballots counting for the candidate with a surplus. Only continuing candidates (those candidates not yet elected or eliminated) can receive surplus votes.

There are two main methods of surplus transfer:

a. Whole ballot transfers, where whole votes (each counting for one vote) transfer to a different candidate, instead of counting for the candidate with a surplus, until the candidate with a surplus has only as many votes as required by the threshold; or,

b. Fractional transfers, where a fraction of every vote counting for a candidate with a surplus is transferred to later-ranked candidates on those ballots, until the candidate with a surplus has only as many votes as required by the threshold.

Whole ballot transfers are currently used in Cambridge, MA elections. Whole ballot transfers are disfavored because they are not guaranteed to proportionally distribute a candidate’s surplus to the other candidates ranked on each voter’s ballot.

Fractional transfers are currently used in Minneapolis, MN elections using a transfer method known as the "Weighted Inclusive Gregory Method" (WIGM). Here is a brief example showing how that transfer method works:

Take a three-seat election with 100 votes cast. The threshold is 25+, from the formula T≥100/(3+1)+n.

The Threshold is the number of valid ballots (B) divided by the number of seats to be filled (S), plus 1. n, as noted above, is some small amount between zero and one. In this case the threshold could be a fraction of a vote more than the 25 votes in our example.

How do we obtain a fraction of a vote?

First, Calculate a "Surplus Fraction." This is the candidate's extra votes (above the threshold) / the candidate's total votes.

Surplus Fraction (SF) = Candidate's extra votes/Candidate's total votes

e.g.: T = 25+, Candidate's total votes = 31

SF = (31-25)/31 = 6/31 = .1935[*]

With a threshold of 25 votes and a candidate who received 31 votes, the surplus fraction would be 6/31 or .1935.

This becomes the "Transfer Value" to apply to each ballot's share of the surplus.

Next, each of the 31 voters' next highest-ranked choice receives .1935 of a vote as the "transfer value" of the surplus.

This "transfers" 5.9985 (31 x .1935) votes to other candidates, leaving the winning candidate with 25.0015 votes, which is >25 or the Threshold.

No preferences are wasted, the order of counting is not a factor and each voter's ballot shares equally in the surplus.

If the last seat(s) in a multi-winner contest is filled and a vote surplus exists, this surplus can be redistributed, leaving all candidates with vote totals equal to the threshold. In general, neither the size of a vote surplus nor the order in which candidates

are declared elected has any substantive meaning in multi-winner elections. What is important is which candidates achieve the threshold number of votes.

In our example, 25.0001 would be the smallest number ≥ to 25, thus, all winners would receive 25.0001 votes and any remainder would be a "residual surplus" (not counting for any candidate) due to rounding.

There are other methods of fractional surplus transfer, but none are used in the United States, so we do not cover them here. All international uses of multi-winner RCV (used in Australia, Ireland, Malta, New Zealand, Northern Ireland, and Scotland) use fractional transfers.

10. How many surpluses to transfer at once (multi-winner RCV)

a. Transfer one surplus at a time (used in Cambridge, Eastpointe, Minneapolis)

i. Setting in RCTab: Winner Election Mode: Multi-winner allow only one winner per round

b. Transfer all possible surpluses at once (referenced in California legislation CA SB 1288 2017, CA SB 212 2019)

i. Setting in RCTab: Winner Election Mode: Multi-winner allow multiple winners per round

It is possible for multiple candidates to cross the threshold of election in a single round in multi-winner ranked choice voting. If that occurs, there are two questions to answer: how many surpluses can transfer at once? And, if only one surplus can transfer, which one transfers?

There are two options for transferring surpluses: transfer just one surplus or transfer all available surpluses. If just one surplus will transfer, then the largest surplus (the candidate with the most votes) transfers. If multiple surpluses will transfer, then all surplus votes transfer at once based on each individual winning candidate's transfer value (as described in (9) above). Votes transferring from winning candidates can transfer only to candidates still in the contest – if a winning candidate (a candidate transferring surplus or a candidate elected in a previous round) is ranked on a ballot, that ranking will be skipped in any surplus transfer. Eliminated candidates also cannot receive surplus votes.

If, when transferring one surplus at a time, two or more candidates cross the threshold with a surplus and have surpluses of the same size, a tiebreaker will be needed to determine whose surplus transfers first. The tie-breaking protocols laid out above in (6) and (7) are sufficient here.

11. Miscellaneous rules/notes

a. Precinct Tabulation

i. Setting in RCTab: Tabulate by Precinct

b. Decimal places for vote counting

i. Setting in RCTab: Decimal places for vote counting

c. Number of rankings on a ballot

i. Setting in RCTab: Maximum number of candidates that can be ranked

2.24.1 Precinct Tabulation

Precinct-level results in ranked choice voting elections are similar to precinct results in plurality elections: they show how many votes each candidate got in a given precinct. RCV precinct-level results differ from plurality results in two important ways, however:

1. They are calculated round by round;
2. They depend on jurisdiction-wide results.

First: as with full results from a ranked choice voting election, precinct-level ranked choice voting results must show the results of each round of counting. Reporting precinct-level results can be done using reporting methods similar to those used for the full RCV count, with a spreadsheet breaking down each round of counting. Examples of this are included later in this discussion. More detailed visualizations have been produced as well.

Second: ranked choice voting is calculated on a jurisdiction-wide level. In practice, that means precinct-level results in RCV provide somewhat different information from precinct-level results in a plurality election.

For example, in a plurality election with three candidates (Candidates A, B, and C), 1000 votes are cast. Candidate A receives 400 votes, Candidate B receives 350 votes, and Candidate C receives 250 votes. To illustrate:

Election Results	Vote Totals
Candidate A	400
Candidate B	350
Candidate C	250
Total Votes	1000

Candidate A wins the election. Candidates A, B, and C may win different precincts, however.

For example, there may be a precinct where Candidate C is very popular:

Precinct 3 Results	Vote Totals
Candidate A	20
Candidate B	30
Candidate C	50
Total Votes in Precinct	100

We generally say that candidate C lost the contest but won this precinct.

Compare this with an RCV election with this same set of candidates (A, B, and C). Again, 1000 votes are cast. Candidate A receives 400 first choices, B receives 350 first choices, and C receives 250 first choices. Because Candidate C has the fewest first choices, C is eliminated. Their votes then transfer to the next candidate ranked on each ballot. Candidate A receives 150 votes and Candidate B receives 100 votes. Candidate A wins with 550 votes and Candidate B winds up with 450 votes. To illustrate:

Round-By-Round Results	Round 1	Transfer	Round 2
Candidate A	400	+150	550
Candidate B	350	+100	450
Candidate C	250	-250	0

There may be a precinct where Candidate C leads in first choices. In this precinct, Candidate C may have 50 first choices, Candidate B may have 30, and Candidate A may have 20. Candidate C may lead in the precinct but, because C has the fewest votes in the contest as a whole, they are eliminated, and their votes transfer.

Precinct 3 Round-By-Round Results	Round 1	Transfer	Round 2
Candidate A	20	+35	55
Candidate B	30	+15	45
Candidate C	50	-50	0
Total votes in Precinct	100		100

In ranked choice voting, Candidate C led this precinct in the first round but, because they lost the contest, was eliminated and had their votes transfer. Because they were eliminated and their votes transferred, Candidate A wound up winning this precinct.

2.24.2 Decimal Places

The number of decimal places to use for voting counting is a detailed, granular decision relevant only to proportional RCV elections. Setting this makes it possible to hand count proportional RCV elections, because the fractions involved in multi-winner RCV can become unwieldy for a human to manage. All RCV jurisdictions in the US that have made this decision have set the number of decimal places at 4. It may be possible to remove the decimal place limitation on proportional RCV elections, but that would make those elections more challenging to replicate in a hand count.

The rounding required to stick to the number of decimal places set for a given multi-winner RCV election can mean that the/a vote value is rounded away. That vote value is recorded as "residual votes" in multi-winner RCV elections in RCTab.

2.24.3 Rankings on the Ballot

The number of rankings on an RCV ballot varies across the United States. The most rankings permitted in an RCV election in the United States is 27 (the 2017 Cambridge City Council contest). Most RCV jurisdictions in the US allow voters to rank between 3 and 10 candidates. These ranking limitations may have an impact on how many votes to exhaust - if there are many candidates in a contest and voters have a limited number of rankings, it is likely that more ballots will run out of candidates as more candidates are eliminated in the round-by-round count. More rankings likely means fewer exhausted ballots.

2.24.4 Glossary of Terms

Glossary of Terms	
batch elimination	A simultaneous defeat of multiple continuing contest options for which it is mathematically impossible to be elected or to prevail. **Discussion** Rule (eliminate the set of one or more candidates C1, C2, C3….Cn if sum of all votes for C1….Cn < votes for candidate B and votes for B < votes for candidate) If a specified candidate satisfies both of the following conditions, then all candidates with fewer votes may be designated as defeated: 1. At least one other candidate has at least as many votes as the specified candidate. 2. The specified candidate has more votes than the total votes for all candidates with fewer votes.
continuing ballot	A ballot that will be processed in the current contest RCV round.
continuing contest option	A qualified candidate, measure, issue, or other contest option that has not yet been elected, approved, or eliminated in the current round or instance of RCV contest processing.
exhausted ballot	A ballot encountered in a round of RCV processing of a contest that has no further valid rankings of continuing contest options or that contains a condition in a subsequent choice that invalidates further consideration of the ballot. Equivalent term: Inactive Ballot **Discussion** The term ‘exhausted ballot’ is the term in use and in practice, in legislation and other authoritative elections official and election administrator documents including RFPs, RFIs, manuals, and documentation as well as in use in the discourse in the field. Therefore, this term is preserved to achieve the purpose, goals, and benefits of this standard. In Multi-Pass IRV all ballots, including exhausted ballots, are revived when beginning the count to fill a second or later seat.
highest-ranked continuing contest option	The next preferred continuing contest option on a given ranked ballot.
highest continuing ranking	The ranking on a voter's ballot with the lowest numerical value or the next highest position in sequence for a continuing RCV contest option.
inactive ballot	Equivalent to Exhausted Ballot. **Discussion** See also: Discussion for Exhausted Ballot. The term “Exhausted Ballot” is used extensively as a term of art in the RCV legislation, local jurisdiction artifacts, contracts, RFPs, and therefore in the field. Therefore, for the purpose of NIST SP 1500-107 we preserve this as a key existing label for the same concept as the newly and uniquely coined Inactive Ballot term.
majority of votes	Greater than 50-percent of the votes counted for a contest for all continuing contest options in an RCV round. **Technical Definition** majorityOfVotes = (sum(aggregate(votesContinuing)) / 2) + 1

Glossary of Terms	
	Discussion “Majority of votes” shall mean fifty percent (50%) plus one of the votes cast on continuing ballots. (San Leandro)
mathematically impossible to be elected or prevail	A contest option in an RCV contest where the current vote total plus all votes that could possibly be transferred to it in future rounds would not be sufficient to surpass the contest option with the next higher current vote total. **Discussion** See Batch Elimination for operation.
overvote	Occurs when the number of selections made by a voter in a contest is more than the maximum number allowed. **Discussion** As applied to RCV, overvotes may result in a skipped selection rather than a lost vote. The consequences of the overvote depend on the RCV rule being applied.
overvote RCV ranking	A ranking assigned to more than one contest option.
ranking, RCV	The number or position selected for a contest option to indicate a voter's ranked preference for that option.
repeated ranking	Selection of more than one ranking for the same contest option for the contest being counted. **Discussion** Sometimes referred to as Duplicate Ranking. Not to be confused with "duplicate ranking" when that term is used to refer to an overvote.
residential surplus	Any vote value lost due to rounding/truncation during the surplus transfer process.
round of counting or round	A round is a subprocess in the tabulation process during which current votes for all continuing contest options are counted in accordance with the applicable tabulation method counting rules. **Technical Definition** A round is a subprocess in the tabulation counting process during which current votes for all continuing contest options are counted in accordance with the applicable tabulation method counting rules. A round is for the purpose of determining the following: • whether a contest option has achieved a majority or a threshold, • whether and which contest option or contest options are to be eliminated. • redistribution of surplus in a multi-winner contest. **Discussion** A round is for the purpose of determining the following: • whether a contest option has achieved a majority or a threshold; • whether and which contest option or contest options are to be eliminated; or • redistribution of surplus in a multi-winner contest.
skipped ranking, RCV	When a voter omits a ranking and ranks a contest option at a subsequent ranking.

Glossary of Terms	
surplus	The number of votes cast for a contest option in excess of the number required to meet or exceed the applicable threshold rule. **Discussion** Applies to multi-winner RCV, aka STV
surplus fraction	The proportion of each vote to be transferred when a surplus is transferred. **Technical Definition** The quotient, rounded down to n decimal places, of a contest option’s surplus divided by the total number of votes for the contest option in the round in which the surplus occurs. **Discussion** Applies to multi-winner RCV, aka STV
threshold/quota, RCV	The number of votes that are sufficient for a contest option to prevail. In STV, it is also the point at or above which additional votes for a contest option are considered to be surplus.
transfer value	The transfer value of a ballot is the one vote or portion of a vote that the ballot will contribute to the vote total for the ballot’s highest-ranked continuing contest option after a surplus transfer or elimination of one or more contest options. **Discussion** When used in surplus transfer, the product, rounded down to n decimal places, of a ballot’s value multiplied by a contest option’s surplus fraction.

2.25 Section 20 - Process Ranked Choice Voting Contest

2.25.1 Overview

Ranked Choice Voting (RCV) is a voting method where the voter is able to identify their candidate choices in order of preference. Although voting the ballot is the same, there are many variations used in processing ballot selections to determine the candidates that are elected. This section will address the methods currently in use in the United States which fall into two major categories, single-winner RCV and multi-winner RCV. Single-winner RCV is also known as Instant Runoff Voting (IRV). Multi-winner RCV is also known as Single Transferable Vote (STV). Both use multi-round processing methods to determine the winner(s) and require Cast Voter Records (CVRs) containing all voter choices for RCV contests to be processed. These methods are most commonly used in jurisdictions where current law requires a winning candidate to have a majority of votes cast (i.e., 50% +1 in a single seat contest) to avoid the expensive alternative of holding a separate primary or runoff election.

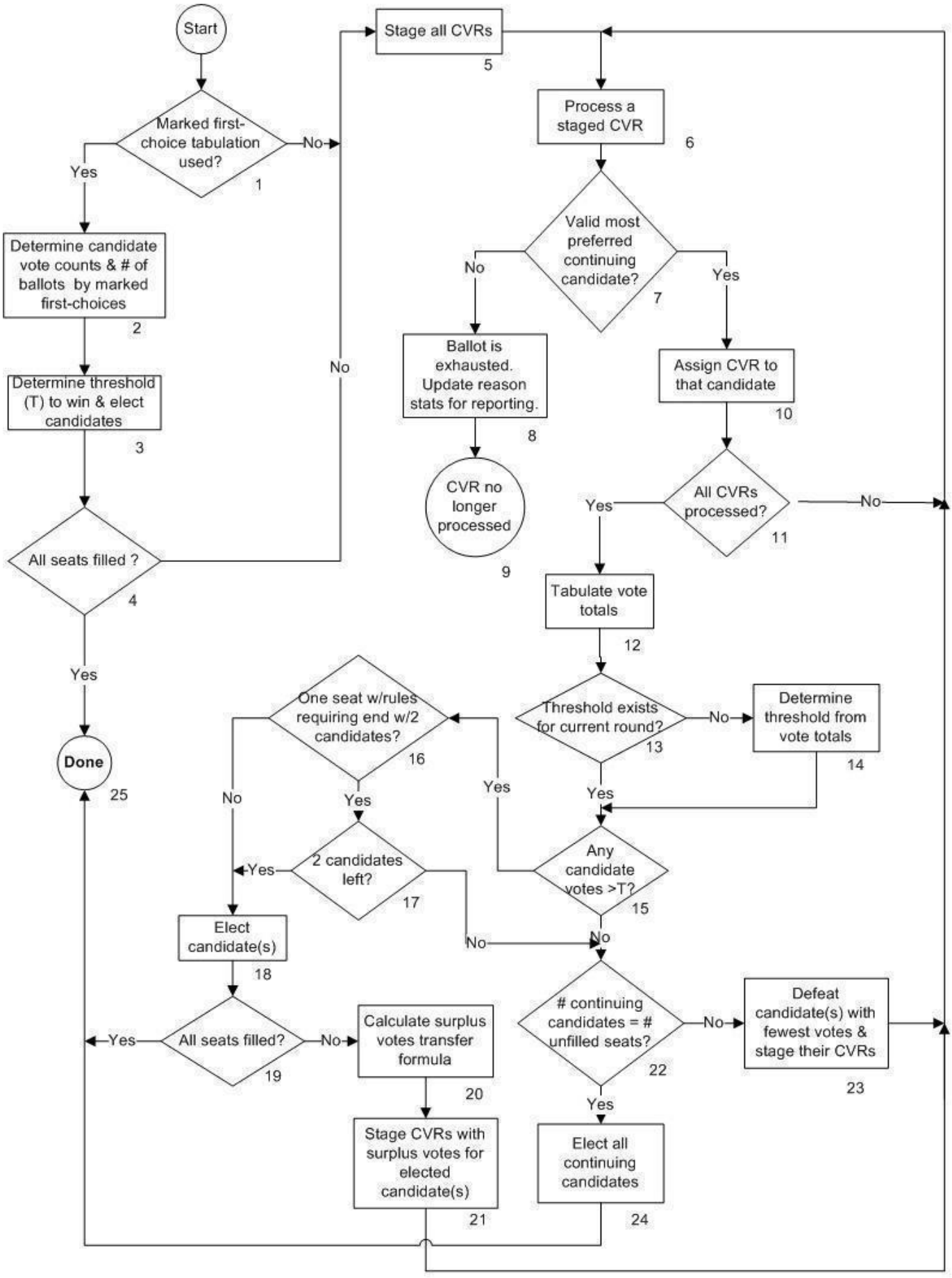
The processing used in each round, if a winner has not been determined, is as follows. The candidate with the lowest vote total on the round is considered eliminated. Each CVR containing the eliminated candidate as the current 1st choice is processed to substitute the next highest ranked continuing candidate (one that has not been eliminated) to replace the eliminated candidate. If there are no choices left on a given ballot, that ballot is considered exhausted. New round totals are tabulated and a determination of whether a candidate now has sufficient votes to win is made. If not, the process is repeated round by round until a winner is or winners are determined.

There are several variations of each of the IRV and STV methods. Most are common but a couple are specific to STV. Common variances include:

- Handling of an overvote choice during the round-by-round processing – Is the ballot considered exhausted or is the choice skipped and checked for a subsequent go forward candidate?
- Handling of an omitted choice/ranking during the round-by-round processing – Is the choice skipped, does this cause the ballot to be exhausted or is the choice skipped once but two skipped rankings in succession exhaust the ballot?
- Handling of a duplicate ranking (selecting the same candidate for more than one ranking) during the round-by-round processing – Is the duplicate skipped in the same way as an omitted ranking or is the ballot considered exhausted?
- Handling of candidate elimination – Are all candidates who have no chance of

  winning concurrently eliminated in the first round only or in any round, or is only one candidate eliminated in any round and multiple elimination of candidates not used?
- Handling of tabulated voted 1st choices – Is tabulation of voted 1st choices used to determine if a candidate(s) has sufficient votes to be elected (thus avoiding the use of

  an RCV algorithm) or is the algorithm always used and tabulation of marked first choices ignored? Even if used, does calculation of the threshold for election include all ballots cast (include over and under vote totals) or is it based on total selection.



This flow chart indicates the processing flow for either single or a multi-seat contest with each block in the flow described.

2.25.2 Description

1. **Marked first-choice tabulation used?**

   Some RCV rules require (St. Paul and Minneapolis) or conditionally allow (Oakland) a shortcut that can elect candidates by looking only at marked first choices. This can help avoid what may be a more logistically involved process of collecting and examining CVRs to determine effective first choices. If a tabulation of marked first choices elects sufficient candidates, the full RCV tabulation process is not needed and examination of CVRs might be avoided. Some other jurisdictions use this shortcut as a matter of practice.

   If a ballot has a validly marked first choice, that is also the ballot's effective first choice. However, a ballot can have an effective first choice but not a marked first choice if the first choice is left blank or is not valid but there is a second or subsequent choice that is validly marked. Detailed rules for what is counted as an effective first choice can vary by jurisdiction. See the description for block 6 for additional details.

   The Yes path is taken if an initial tabulation of marked first choices is used. The No path is taken if the CVRs are used to initially determine the effective first-choice candidate votes.

2. **Determine candidate vote counts & # of ballots by marked first-choices**

   Votes are tabulated by considering only marked first-choice ballot selections and as if the contest were a vote-for-one contest. If the first choice is not marked or is overvoted on a given ballot, there will not be any contribution to candidate votes. The number of ballots used as the base for the threshold calculation is also determined based on jurisdiction-specific rules or practice. St. Paul uses the total number of ballots with validly marked first choices that count for candidates. Minneapolis uses the number of cast votes. The Oakland rule uses the number of ballots cast except for those with a marked overvote.

3. **Determine threshold (T) to win and elect candidates**

   The threshold identifies how many votes a candidate must have in order to be elected based by the tabulation of marked first choices. If a contest is only electing one candidate, the threshold is typically a majority of the threshold base so that a candidate would need more than 50% of the threshold base to be elected. The Minneapolis threshold formula for any number of candidates to be elected is:

   T= 1 + B / (N + 1) rounded down to the nearest whole number

   where `T` is the threshold, `B` is the number of relevant ballots and `N` is the number of candidates to be elected.

   Thus, in a contest where only one candidate is to be elected, the Minneapolis threshold is floor (1 + B / 2) and a candidate must have at least that many votes to be elected. In a contest with 4 candidates to be elected, the Minneapolis threshold would be floor(1 + B / 5) and a candidate must have at least that many marked first-choice votes to be elected, i.e., more than 20%, since the number of marked first-choice votes will be a whole number. Once the threshold and threshold criterion are established, any candidates satisfying that criterion are considered elected for the purposes of this tabulation of marked first choices.

4. **All seats filled?**

   Are sufficient candidates elected because their number of marked first-choice votes is greater than (or equal to, for Minneapolis) the threshold to fill all seats in the contest? The Yes path is taken if there are. The No path is taken if one or more positions have not been filled due to insufficient candidates obtaining the required number of marked first-choice votes. In a single-winner contest, taking the Yes path only requires that one candidate be elected. If the No path is taken, a full RCV tabulation is conducted, without using the results of the tabulation of marked first choices. In particular, candidates elected by marked first choices are considered unelected continuing candidates at the beginning of the full RCV tabulation. If the Yes path is taken, the full RCV tabulation is not required.

5. **Stage all CVRs**

   A full RCV tabulation requires every CVR to be examined to determine which if any candidate the CVR will count for in the first round. This block stages all CVRs for that determination. All candidates begin the first round as continuing candidates, neither elected nor eliminated, without any votes.

6. **Process a staged CVR**

A single CVR taken from a collection of staged CVRs is processed to determine the candidate for whom the ballot will next count, the CVR's highest ranked continuing candidate, if such a candidate exists. A continuing candidate is defined as a candidate that is neither eliminated (a.k.a defeated) nor elected.

CVRs are staged for processing in this step from three sources:

a. in block 5, all CVRs are staged for round 1,

b. in block 20 after a candidate has been defeated, and

c. for multi-seat contests in block 18 after a candidate has been elected with surplus votes, the votes for a candidate in excess of the threshold.

Minneapolis rules, CA SB 1288, and HR 3057 redistribute surplus as a fractional vote for all CVRs that counted for the elected candidate. Cambridge rules distribute surplus as a whole vote per CVR but only for a subset of CVRs. See the description of block 18 for further details. Typically, the CVR's counting for a candidate is staged for reassignment after that candidate is eliminated or elected and before the next round's vote counts are tallied, unless it can be otherwise determined that a next round is not needed.

Determining the highest-ranked (most preferred) continuing candidate can be fairly straightforward if the voter has simply ranked candidates in order of preference. However, there are several irregular ranking situations that a voter might mark and which are treated according to jurisdiction-specific rules:

a. no candidate ranked at a ranking level,

b. ranking more than one candidate at a ranking level, and

c. ranking a candidate at more than one ranking level.

The following describes how these situations are treated by various jurisdictions:

- Unvoted choice (no candidate ranked at a ranking level)
- Ballot considered exhausted
- Ballot considered exhausted if there are two unvoted choices in succession
- Skipped and subsequent choice processed, if any
- Overvoted choice (more than one candidate ranked at a ranking level)
- Ballot considered exhausted (most common)
- Not considered overvoted if doesn't contain more than one continuing candidate (i.e., Takoma Park, MD)
- Skipped and subsequent choice processed, if any
- Repeated ranking of a previously ranked candidate
- Ballot considered exhausted
- Skipped and subsequent choice processed, if any

7. **Valid most preferred continuing candidate?**

   The No path is taken for the processed CVR when there is no valid selection for the highest ranked continuing candidate. An invalid choice could include selections for more than one continuing candidate depending on jurisdiction rules. The Yes path is taken if there is a valid selection for the highest ranked continuing candidate.

8. **Ballot is exhausted. Update reason stats for reporting**

   This CVR will not be included in any further tabulation processing as there are no more validly ranked continuing candidates available. This may be due to not containing further ranking selections, an overvote in the current ranking choice preventing consideration of subsequent rankings, the repeated selection of an already ranked candidate, or that all subsequent ranking selections are for eliminated or already elected candidates. The reason for the ballot being considered exhausted is recorded for purposes of reporting round results.

9. **CVR no longer processed**

   Since the CVR/ballot does not contain a valid subsequent choice for the highest ranked continuing candidate, that ballot is not subject to further tabulation.

10. **Assign CVR to that candidate**

    The CVR is assigned to count for its highest-ranked (most preferred) continuing candidate. The ballot will contribute one full vote or a surplus fraction of a vote to that candidate's vote total. A surplus fraction of a vote can be used in a multi-seat contest using rules from Minneapolis, SB 1288, or HR 3057.

11. **All CVRs processed?**

    The No path is taken to process the next staged CVR if some staged ballots still remain to be processed. The Yes path is taken if all staged ballots have been processed.

12. **Tabulate vote totals**

    The vote totals for each candidate and for any other reporting categories are tallied.

    If there was a tabulation of marked first choices in block 2 (a.k.a round 0), the candidate vote totals for round 1 can be higher, but never lower, than the candidate totals of marked first choices. A candidate vote total can be higher if there are one or more ballots with no marked first choice, or in some jurisdictions an invalidly marked first choice, but there is a valid candidate selection for a subsequent choice.

13. **Threshold exists for current round?**

    The No path is typically taken every time for a single-seat contest as the majority threshold will be calculated for each round. The No path is typically taken only for the first round for multi-seat contests, so that the same threshold applies to all elected candidates, regardless of the round in which they are elected. The Yes path is typically only taken for the second and subsequent rounds of a multi-seat contest which reuses the first-round threshold. Typically, a threshold from a tabulation of marked first choices will not be reused here, especially if that tabulation used a different threshold base.

14. **Determine threshold from vote totals**

    The total votes counting for candidates in the round is typically used as the threshold base, i.e., the number of ballots / votes used to calculate the threshold. For single-seat contests, the threshold is typically expressed in terms of a majority of that threshold base, i.e., more than 50%. For multi-seat contests, the threshold can be expressed as:

    $T = B / (S + 1) + X$

    where `T` is the threshold, `B` is the threshold base, `S` is the number of seats to be filled, and `X` is some small extra amount that, depending on the specific rules, might be as small as zero but is not bigger than one whole vote.

    There are two approaches to determining the extra amount X and the threshold criterion in order to ensure that it is mathematically impossible to elect too many candidates:

- `X` must be greater than zero, but reaching the threshold is sufficient to be elected
- `X` can be zero, but the threshold must be exceeded in order to be elected

    Minneapolis and Cambridge rules use the first, more traditional approach while the more recent rules in SB 1288 and HR 3057 use the second approach. For example, Minneapolis rules describe `X` as being the result of adding one and ignoring any fractional value, i.e., rounding down to the nearest whole number. SB 1288 rounds up to the fifth decimal place, its precision for calculating fractional votes. In the diagram, the threshold criterion is expressed in terms of the second approach with the understanding that the greater-than-or-equal-to criterion of the first approach can be substituted as appropriate.

15. **Any candidate votes > `T`?**

    The Yes path is typically taken if the vote total for any (continuing) candidate satisfies the threshold criterion, i.e., is greater than the threshold. In a multi-seat election, it is possible for more than one continuing candidate to exceed the threshold in a round. The No path is taken if there are no continuing candidates with a vote total that satisfies the threshold criterion.

    Minneapolis has an exceptional rule that requires, subject to defined conditions, that the No path to be taken in order to eliminate one or more candidates, even if one or more continuing candidates has enough votes to satisfy the threshold criterion. CA SB 1288 has a default provision and HR 3057 requires that the No path be taken for single-seat contests as long as there are three or more continuing candidates. This can extend the tabulation to show a one- on-one comparison between the two finalists without changing which candidate is elected. San Francisco has adopted this option in practice.

16. **One seat with rules requiring end with 2 candidates?**

    For a single seat contest, the sum of existing RCV users require that the recursive process of candidate elimination and promotion of the subsequent highest-ranking continuing candidate continue until only 2 candidates remain even if a candidate reaches the threshold to be elected. The Yes path is taken if these rules apply. The No path is taken if the contest is either a multi-seat contest or the conventional rules are used for a single seat contest.

17. **2 candidates left?**

    This block is reached if the rules for single seat contests require the RCV process to continue until 2 candidates are left. The Yes path is taken if there are only 2 candidates left and the winner will be declared. The No path is taken if there are more than 2 candidates left and cause the RCV process to continue.

18. **Elect candidate(s)**

    One or more of the continuing candidates with a vote total that satisfies the threshold criterion are elected. Depending on jurisdiction-specific rules, if there is more than one such candidate, all them might be elected or only one might be selected for being elected in this round, typically the candidate with the most such votes. Jurisdiction-specific rules for resolving a tie for having the most votes may apply. A candidate that satisfies the threshold criterion but is not elected remains a continuing candidate and is still eligible to receive transferred votes from other candidates.

19. **All seats filled?**

    The Yes path is taken if this is a single seat contest or if all required candidates in a multi-seat contest have been elected, indicating the process has been completed. The No path is taken if it is a multi-seat contest and all seats have not been filled.

20. **Calculate surplus votes transfer formula**

    In a multi-seat co**ntest when one or more candidates are elected in a round, but all seats are not filled, CVRs containing excess votes for the elected candidate are staged for further processing along with the other CVRs for continuing candidates. This step determines the formula for how these CVRs are staged.

    There are two methods currently used for handling surplus votes (votes for an elected candidate that are in excess of the threshold). Minneapolis, CA SB 1288, and HR 3057 each select all CVRs for the elected candidate but assign each CVR a transfer vote value that is a fraction of a whole vote that corresponds to the prorated ballot's share of the elected candidate's surplus. The fraction is equal to the ratio of the elected candidate's surplus votes for the round divided by the elected candidate's total votes for that round. Jurisdiction-specific rules may specify the precision and any rounding (typically rounding down) that are associated with this arithmetic operation.

    In contrast, Cambridge processes ballots from precincts in a randomly chosen order and selects every Nth ballot where N is the total votes for the elected candidate divided by the excess votes (rounded) and transfers the full vote of the selected CVRs to continuing candidates.

21. **Stage CVRs with surplus votes for winning candidates**

    All CVRs for continuing candidates are staged including CVRs containing surplus votes for any candidate elected in this round according to the formula developed in Step 20. In Minneapolis, all CVRs will be transferred with a vote value fraction times the CVR's previous transferred vote value. Note that the previous transferred value might be a fraction if it was a surplus from a candidate elected in a previous round. Jurisdiction specific rules may specify the precision and any rounding (typically rounding down) in this multiplication.

    In Cambridge, the CVRs will be selected and staged at full vote value, according to the formula determined in Step 20.

22. **# continuing candidates = # unfilled seats**

    The No path is taken if there are more continuing candidates than the number of unfilled seats. The Yes path is taken if the number of continuing candidates is equal to the current number of unfilled seats.

23. **Defeat candidates with fewest votes and stage their CVRs**

    A common approach is to eliminate (a.k.a. defeat) the candidate with the fewest votes and then transfer that candidate's votes to other candidates. There can be jurisdiction-specific rules for how to resolve a tie for having the fewest votes. Some jurisdictions run a lottery while others look at previous round results and eliminate the candidate with the lowest votes in the most recent previous round that is not tied, using a lottery only if there is still a tie after looking at all previous rounds.

    It is also common for single-seat contests to allow or require a group of candidates with the fewest votes to be eliminated in a single round (a.k.a batch elimination) if their combined vote totals are less than the candidate with the next higher vote total. Use of this option will not change who is elected, i.e., its use is outcome invariant. San Francisco and Oakland rules require use of this option but Alameda County, which administers Oakland's RCV elections, does not use it and San Francisco has stopped using it in favor of other tabulation options its voting system supports.

    Some rules have exceptional elimination rules. For multi-seat contests, Minneapolis and HR 3057 require certain candidate eliminations, including batch eliminations, even though there might be continuing candidates that satisfy the threshold criterion for being elected. Cambridge requires elimination, after any surplus is transferred from candidates elected in the first round of every candidate with fewer than 50 votes. The 50-vote minimum is derived from the requirement to have 50 signatures on a candidate's nominating petition. For single-seat contests, Minneapolis has a rule requiring elimination of a candidate based on the total number ballots on which a candidate is ranked. None of these exceptional elimination rules is guaranteed to be outcome invariant compared to single elimination only when there is no surplus to be transferred.

24. **Elect all continuing candidates**

All continuing candidates are elected in order to fill the remaining unfilled seats. This allows a candidate to be elected without satisfying the threshold criterion.

25. **Done**

    Indicates that candidates have been elected to all positions to be filled and the tabulation process is complete.

2.26 Section 21 - Ballot Limitations & Maximum Testing Range

All vendors have ballot limits for using Ranked Choice Voting. You should check with your voting system vendor for ballot layout limitations. See the list below for the five major vendors in the US and their Ranked Choice Voting limitations.

- ES&S
- Portrait: 21 Maximum Number of Rankings
- Landscape: 23 Maximum Number of Rankings
- Unisyn Voting
- Grid/Portrait style: 3 Maximum Number of Rankings
- Hart Intercivic
- Grid/Portrait style: 6 Maximum Number of Rankings
- Dominion Voting
- Grid/Portrait Style: 10 Maximum Number of Rankings
- Clear Ballot
- Not RCV compatible at this time.

We recommend that RCTab be tested at the outer limits of the ballot layout to ensure that the software can tabulate accurately within the range of the ballot limitations.

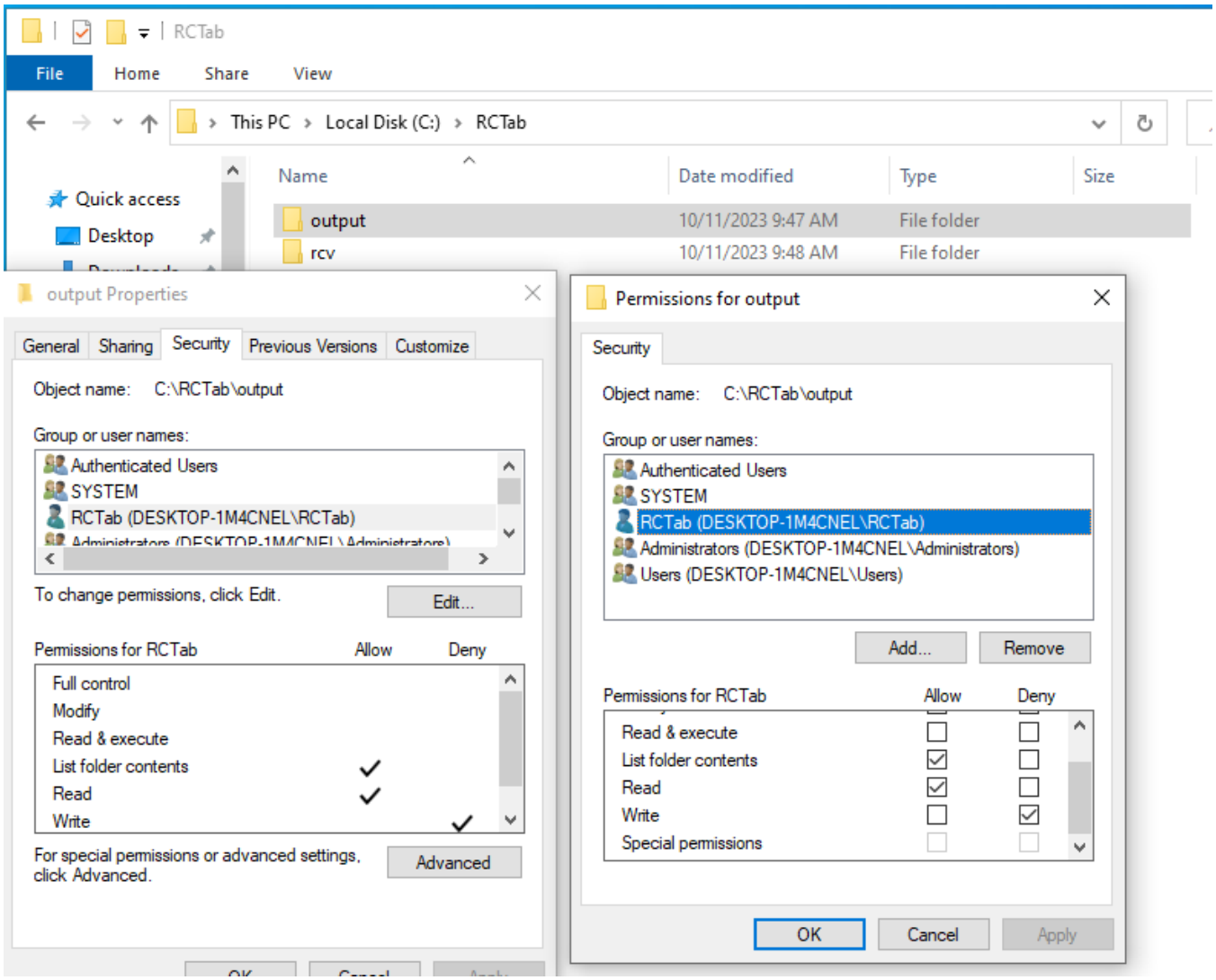
2.27 Section 22 - Installation Instructions for Windows OS

2.27.1 Installing RCTab v2.0

- ***Never connect the internet to the Workstation RCTab is installed on.***
- Only Election Administrators should complete the following installation steps. It requires logging in with a Windows administrator level account created in **Section 16 - System Hardening Procedures - Windows OS**. As described in that section, credentials for the Windows Administrator account should follow secure password practices and should be provided to the least amount of users necessary to complete the following steps. Users who will be running the tabulation shall not have access to the Administrator account.

- Log in to an administrator-level account on the RCTab machine.
- Follow **Section 16 - System Hardening Procedures - Windows OS** before installation of RCTab.
- **Create RCTab Windows Standard User Account** To ensure that RCTab users have the least OS-level permissions necessary to run a tabulation we will create an "RCTab" Windows standard user account. The standard user account has less permissions than a Windows administrator account. Jurisdictions shall use this standard "RCTab" account when running tabulations with RCTab.
- Open Windows System Settings by clicking the Windows Button -> Settings
- Select "Accounts"
- Click "Family & other users" in the left menu list
- Click "Add someone else to this PC"
- Click "I don"t have this person"s sign in information"
- Click "Add a user without a Microsoft Account"
- Enter "RCTab" as the User name
- Enter a secure password
- Enter security questions
- Hit "Next"
- The "RCTab" account has been created. Continue installation instructions as an administrator-level account until installation instructions say to log in as RCTab user
- Contact the relevant authority to request the Trusted Build of the RCTab v1.3.2 software.
- The relevant authority will provide the procedures to receive the Trusted Build to install RCTab.
- Upon receipt of RCTab, plug the flash drive RCTab is saved on into the USB port on the hardware to be used for RCTab installation.
- Using File Explorer, locate the `C:\` drive and open it.
- Right-click in the open space in the right-hand frame and select new and left-click on the folder.
- Name the new folder `RCTab`. This folder is now located at `C:\RCTab`
- Using File Explorer, locate the `rctab_v2.0_windows.zip` file.
- Right-click on the `.zip` file and select copy.
- Using File Explorer, open the RCTab folder.
- Right-click in the open space and select paste.
- To validate that the downloaded zip folder contains the certified version of RCTab, follow the instructions in document **Section 23 - Trusted Build & Output Hash Verification - Windows OS**
- After validating the hash of the trusted build navigate to the RCTab folder and locate the `rctab_v1.3.2_windows.zip` file.
- Right-click on the file. Then click on the tab “Compressed Folder Tools” at the top of the file explorer window. Click on “Extract All” at the top of the File Explorer.
- A window will pop up. Confirm that the extraction location is the RCTab folder created earlier. Click extract.
- The .zip file will now extract. You now have a folder called `rctab_v1.3.2_windows`

- **Enforce Read-Only Permissions on Output folder** To enforce read-only permissions for all RCTab output you **must** set the following permissions on the folder your jurisdiction will use for RCTab output.
- Create a folder to be used for RCTab output. In order for RCTab to write output files as read only **do not use paths within the Windows User folders, like Desktop or Documents (**that is anything under the path `C:\users\`). This requirement is programmed into RCTab - it will not allow the output path to be configured to a Windows user account folder. For these instructions we will create an `output` folder in the `C:\RCTab` folder we created earlier
- Right-click the `C:\RCTab\output` folder and select Properties
- Click the "Security" tab
- Click "Edit" to change permissions
- Click "Add"
- Type "RCTab" in the “Enter the object names to select” text box
- Click the "Check Names" button. If successful, the user name will be underlined. It might include the computer name in front of it. This is ok as long as it is underlined.
- Click "Ok"
- Uncheck "Allow" for "Read and Execute"
- Check "Deny" for "Write"
- Click "Apply"
- Click "Ok"

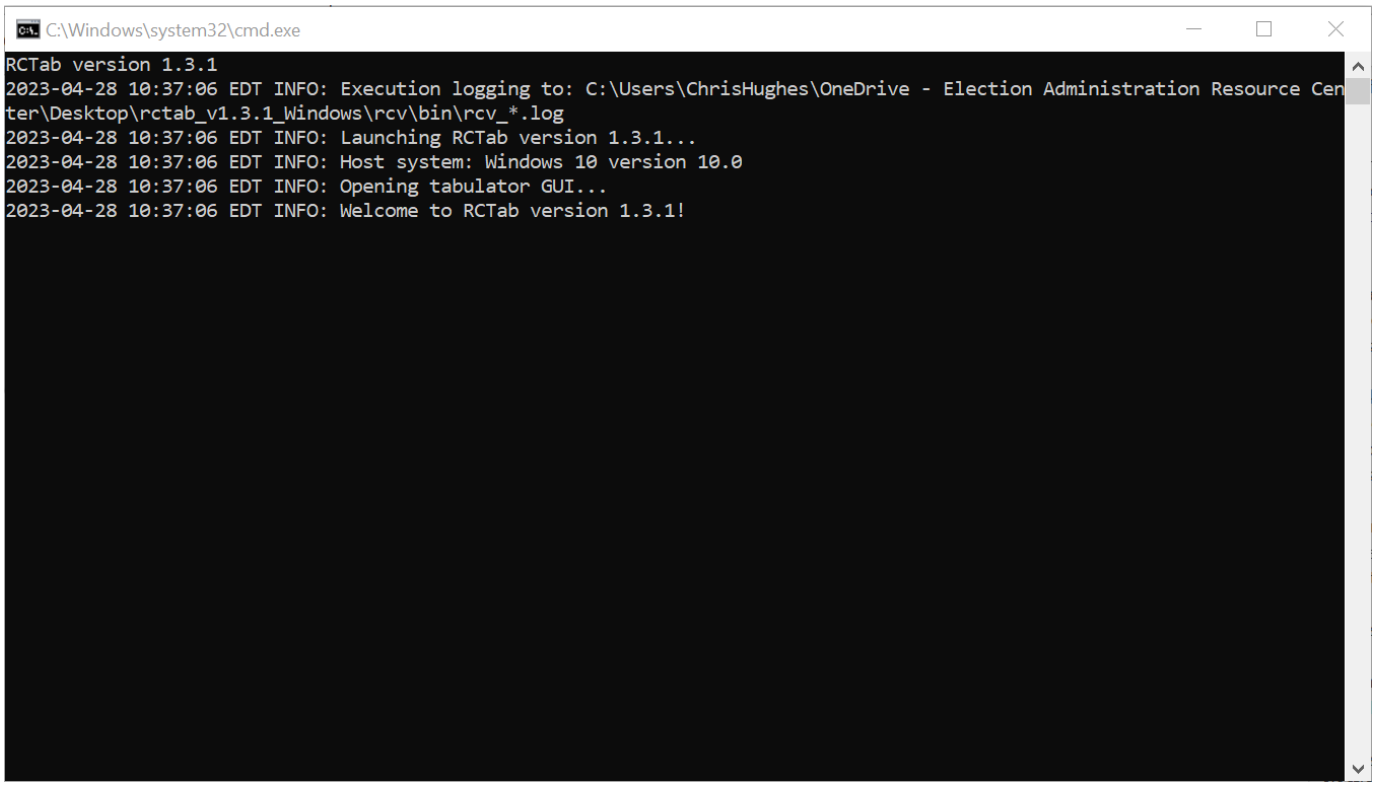


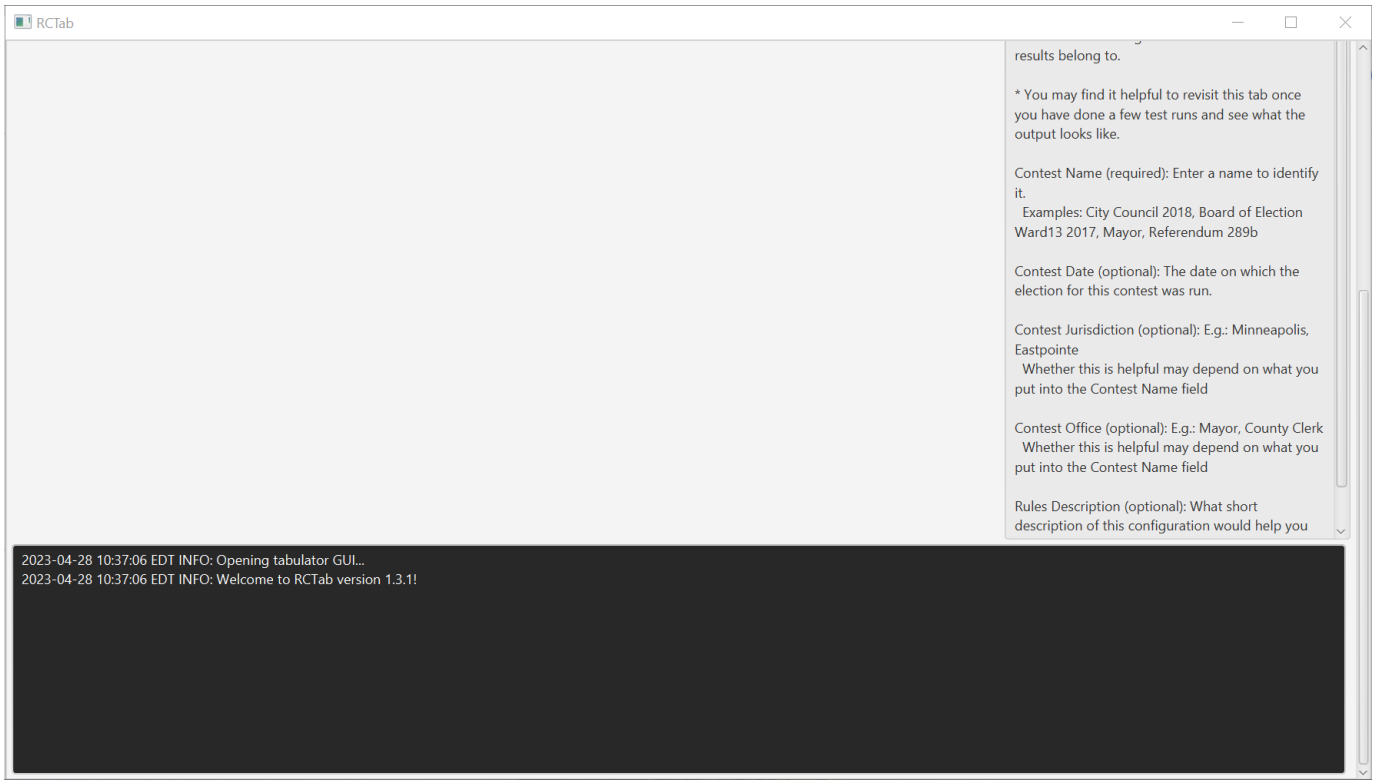
Note: RCTab contest summary output files are programmatically set to be unmodifiable. However, if the folder they are exported to is not set properly with the preceding steps users could *delete* them.

- Now, log in as the "RCTab" Windows standard user. Tabulation should **always be done logged in to the Windows OS as the "RCTab" Windows standard user.**
- Navigate to the `rctab_v2.0_windows` folder. Double-click on the `rcv` folder.
- Double-click on the `bin` folder.
- Right-click on the `rcv.bat` file Click "Run as Administrator." If a "Windows protected your PC" window pops up click "More Info" then click the "Run anyway" button. Enter the administrator password
- RCTab will now launch.
- Using your mouse left click and drag the window with the dark top border down until it looks like the pictures below.
- Check that the first line on the black background shows:

```
RCTab version 2.0
```

- You have successfully installed the RCTab v2.0!
- Optionally, you can create a shortcut on your desktop for a simpler start process in the future
- Right-click the `rcv.bat` file -> Create Shortcut
- Right-click the shortcut file created -> Properties -> Advanced -> Check the "Run as administrator" checkbox -> Ok -> Ok
- Drag the shortcut file to your desktop
- Complete the instructions in **Section 23 - Trusted Build & Output Hash Verification - Windows OS** under the "Validating the hash of a trusted RCTab build" header to verify your installation.





2.28 Section 23 - Trusted Build & Output Hash Verification - Windows OS

2.28.1 Validating the hash of a trusted RCTab build .zip

To validate that the provided zip folder contains the certified version of RCTab v1.3.2, follow these instructions:

- Open the Start Menu
- Type in Command Prompt
- Press "enter" to launch the Command Prompt
- Type in `C:\Windows\System32\certutil.exe -hashfile [filename]`
- To insert the file name: locate the `rctab_v1.3.2_windows.zip` in File Explorer
- Left-click on the file and while continuing to hold the mouse down, drag the file to the Command Prompt window and place after `-hashfile`.
- Example: `C:\Windows\System32\certutil.exe -hashfile C:\RCTab\rctab_v1.3.2_windows.zip`
- Add `SHA512` to the end of the line
- Example: `C:\Windows\Systems32\certutil.exe -hashfile c:\RCTab\rctab_v1.3.2_windows.zip SHA512`
- Press "enter". The command prompt will show the text of the hash in the following format `SHA512 hash of [filePath]: [SHA512 hash text]`
- Compare the `SHA512` hash code produced on your system to the SHA512 hash code supplied with the trusted build.
- If the `SHA512` hash codes match 100%, you are using an approved download and can proceed. Return to the **Section 22 - Installation Instructions for Windows OS** document to complete installation.

2.28.2 Validating the hash of RCTab contest summary files

RCTab automatically creates corresponding `.hash` files for the `summary.csv` and `summary.json` output files. Follow these instructions for using the [fileName].hash files to verify their corresponding summary file. For this example we will verify a `summary.csv` file and we will assume that it is located in `c:\RCTab\output\`

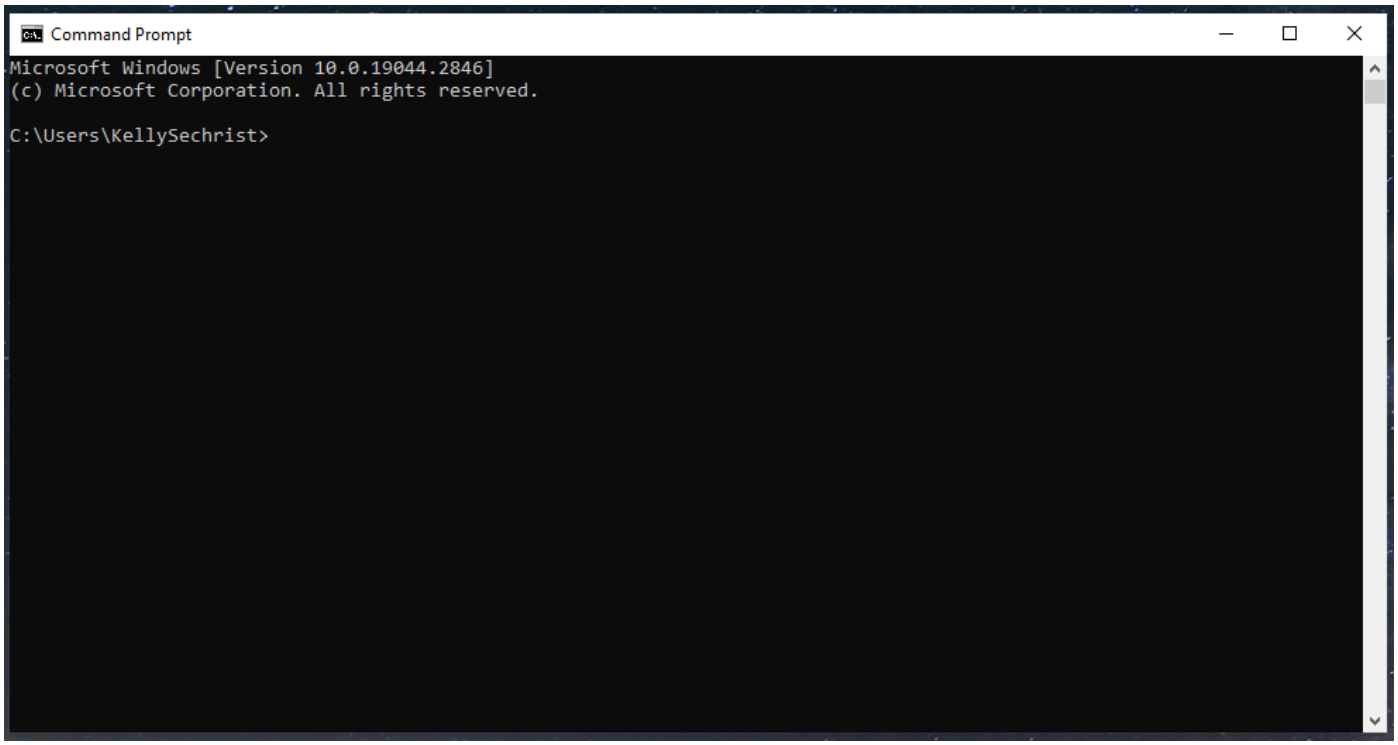
- Create a new empty text file. We'll call this the 'comparison text file'
- Copy the RCTab programmatically generated hash to the comparison text file
- Open `summary.csv.hash` in text editor. This file contains the following format: `[hash] [hashAlgorithm]`
- `[hash]` is the text of the hash itself
- `[hashAlgorithm]` is the algorithm used to get the hash e..g SHA512
- Copy the `[hash]` to the comparison text file on a single line
- Use cmd prompt to generate hash of summary.csv
- Open the Start Menu
- Type in Command Prompt
- Press "enter" to launch the Command Prompt
- Create the command to hash the summary.csv with the following template `C:\Windows\System32\certutil.exe -hashfile [filePath] SHA512`
- Replace `[filePath]` with the location of your summary.csv. You can drag the file into the cmd prompt window to have it automatically fill it in for you or you can type it manually if you know the path
- Press enter. The command prompt will show the text of the hash in the following format `SHA512 hash of [filePath]: [SHA512 hash text]`
- Copy and paste `[SHA512 hash text]` to the next line of the comparison text file
- Compare the text of the two `SHA512` hashes. Pulling the width of the text editor wide enough so that the hashes each are one line each will help line them up for comparison.

2.29 Section 24 - Tabulator Command Line Instructions

To run the tabulation for a config file via the command line, users must open a terminal, navigate to the root directory of the unzipped software build, and use a command similar to the one that launches the GUI, except with different arguments. The final argument is the name of the config file to be tabulated. All tabulation will be done in the terminal window. RCTab's user interface will not appear. If command line launch is successful, RCTab will tabulate a contest according to the requirements of the configuration file in the command line argument. Following are detailed steps for how to launch the RCTab in the command line.

On Windows:

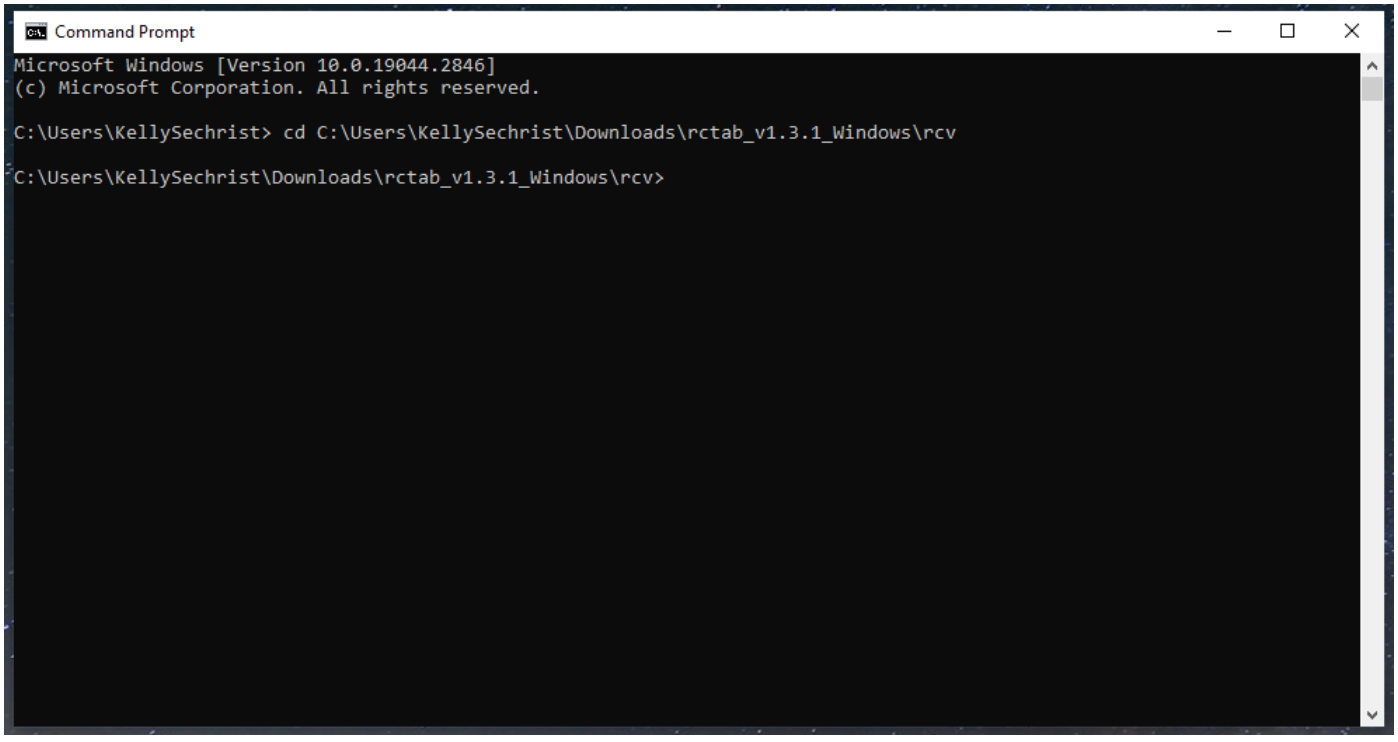
1. Open Start Menu
2. Type Command Prompt. Hit enter.
3. Command Prompt will launch. The screenshot below displays an example of Command Prompt.



4. The user needs to direct Command Prompt to the folder where RCTab is installed. Type cd. Insert the folder name: locate the folder rcv under the folder `rctab_v1.3.1_windows` in File Explorer.
5. Click on the rcv folder and while continuing to hold the button down, drag the file to the Command Prompt window and place after cd.

a. Example: `cd C:\RCTab\rctab_v1.3.1_windows\rcv`

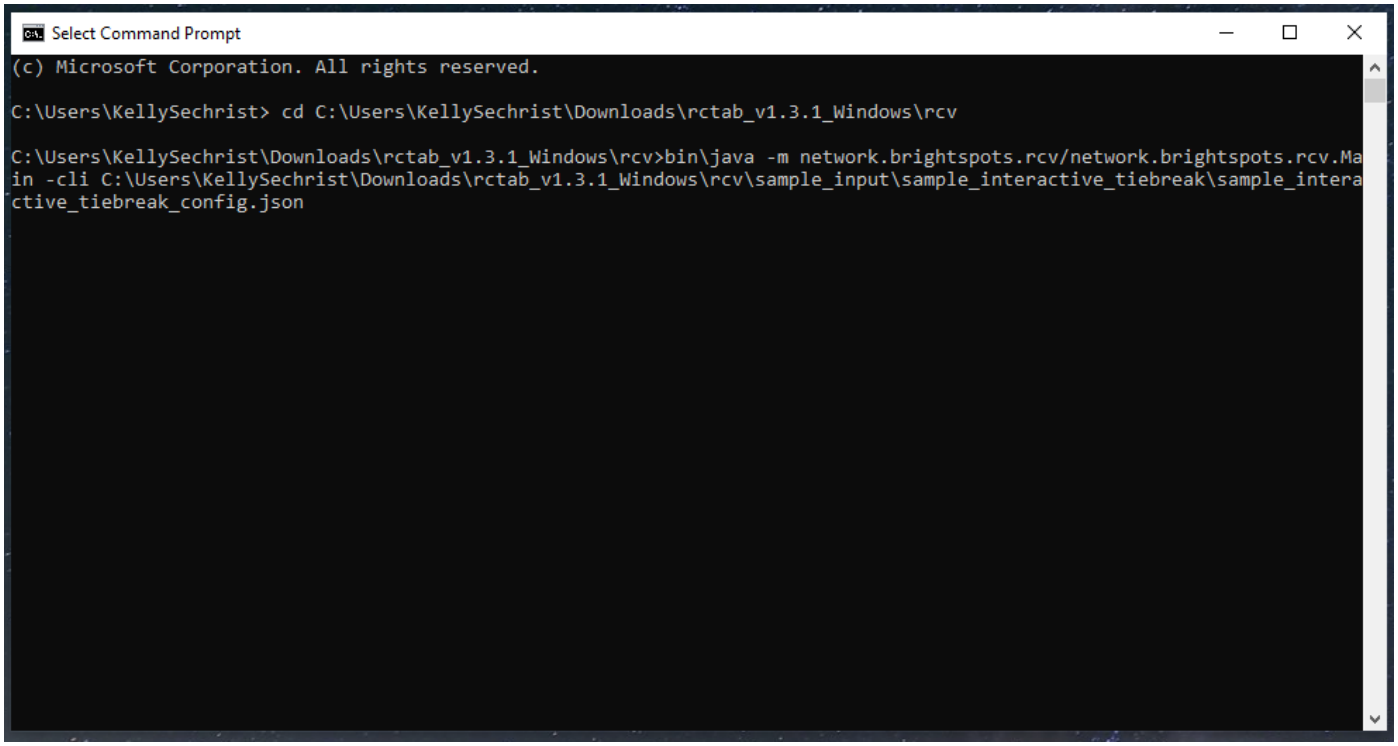
6. Hit enter. The screenshot displays the result of a successful execution of previous steps.



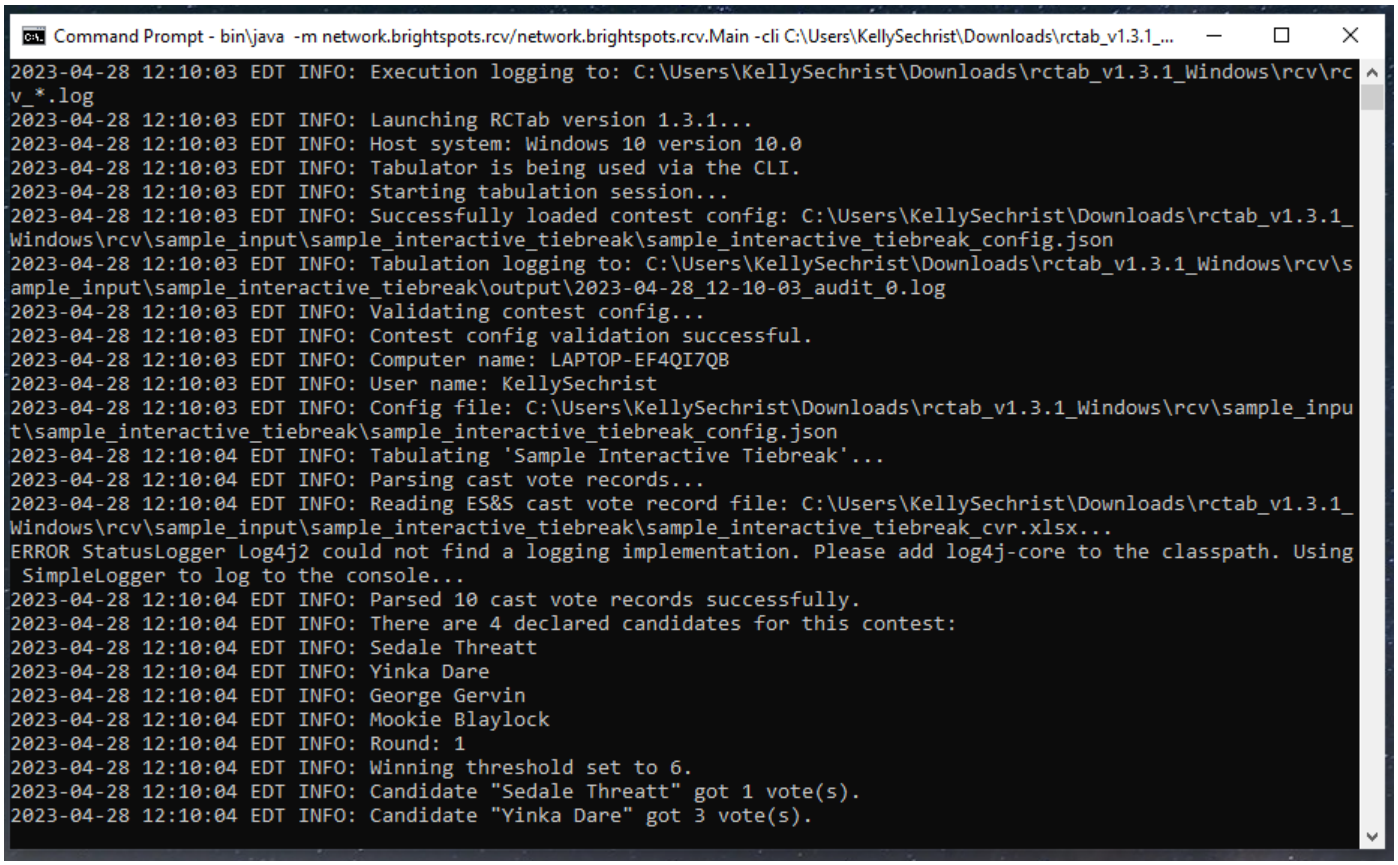
7. Type in `bin\java -m network.brightspots.rcv/network.brightspots.rcv.Main -cli [filename]`

a. Filename should be the name of the configuration file for the RCTab to use to process the contest. Example: `C:\RCTab\rctab_v1.3.1_windows\rcv\sample_input\sample_interactive_tiebreak`

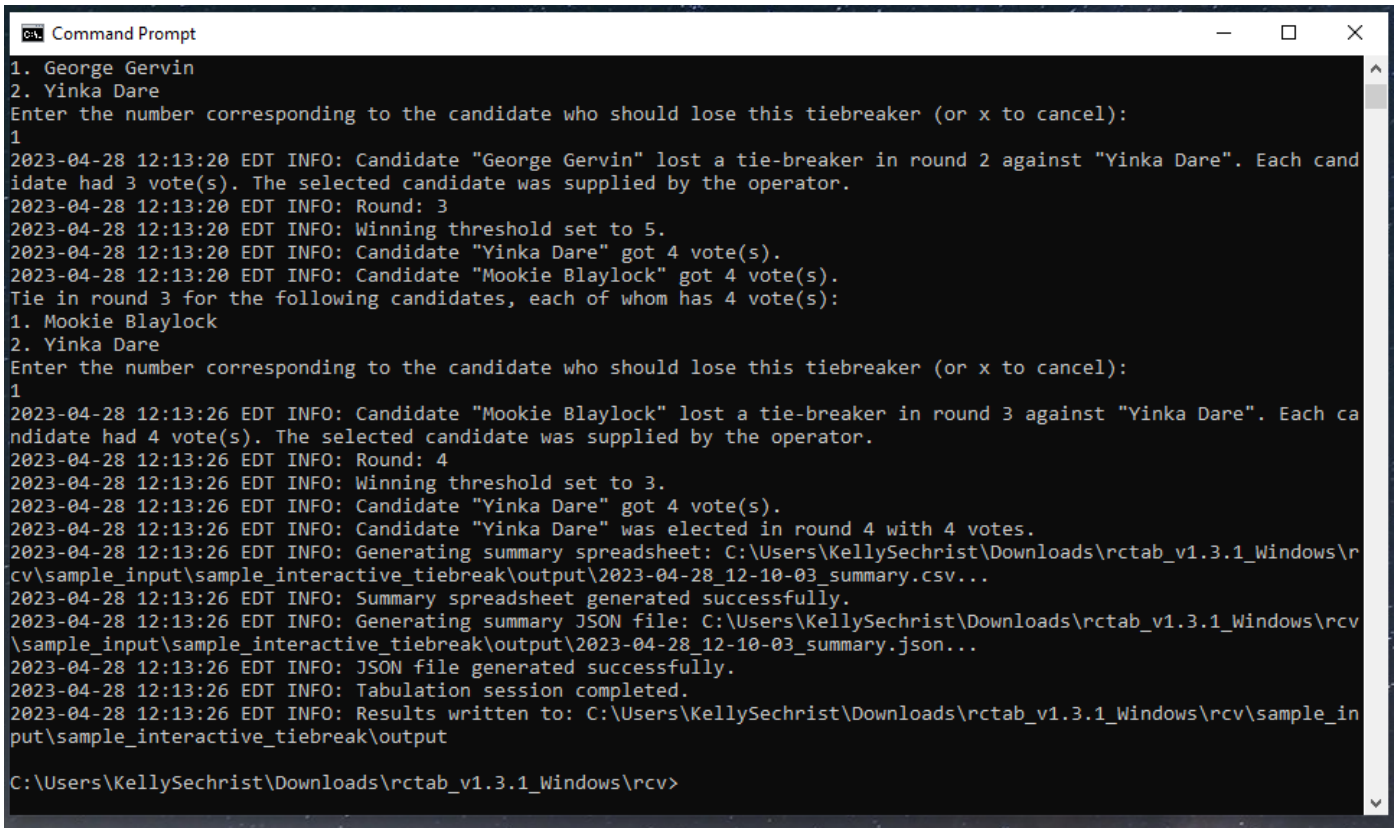
b. Screenshot shows an example of correct command prompt entry for this example.



8. Filename can be copied over into the Command Prompt by finding the configuration file in File Explorer, clicking on the file, and dragging the file over to the Command Prompt window.
9. Press enter.
10. Tabulation will run. Messages explaining the process of tabulation will appear in the Command Prompt window. Screenshot shows an example of messages sent during tabulation.



11. When tabulation is complete, Command Prompt will display the message: "INFO: Tabulation Session Completed." Followed by a line stating, "Results written to: [filepath]". The file path will be based on the information included in the Output Directory setting in the configuration file. See **Section 18 - User Guide** and **Section 25 - Configuration File Parameters** for more information. See below for an example of messages sent at the end of successful tabulation.



12. If any of the above commands do not work, double check that you have copied them all over correctly, that you are pointing the command prompt to the correct directory, and that all filenames are correctly entered.

2.30 Section 25 - Configuration File Parameters

This document lists the parameters the user will configure within RCTab. The output is the JSON file used for tabulation.

The Config file must be a valid JSON format. Example config files can be found under the test folder. The values a user inputs into any of these fields depend upon the relevant laws and regulations in place in their jurisdiction, as well as the voting system vendor used to produce cast-vote records for their elections. Users must understand the requirements of their laws, regulations, and vendor CVR data in order to fill out these fields successfully.

This document lists the parameters that can be included in a config file suitable for input tabulation. Parameters are categorized by required and optional. If a user fails to fill out a required parameter when building a configuration file, RCTab will alert the user to the missing requirement and provide suggested resolution steps.

Config file must be valid JSON format. Examples can be found in the test_data folder.

- `"tabulatorVersion"` required
- version of the application that created this file
- used for migrating config data between different application versions
- example: `"1.0.1"`
- value: text string of length [1..1000]
- `"outputSettings"` required A list of output settings and their associated parameters

  The `"outputSettings"` section contains the following parameters:

a. `"contestName"` required

i. The name of the contest

ii. Used for naming audit output files

iii. Example: `"Portland Mayoral Race 2017"`

iv. Value: text string of length [1..255]

b. The `"outputDirectory"` optional

i. Directory for audit output files (absolute or relative path)

ii. Example: `"/Path/To/TabulatorResults"`

iii. Example: `"output data/contest1"`

iv. Value: string of length [1..255]

v. If not supplied: files will be saved to the current working directory

c. The "contestDate" optional

i. Date of the contest

ii. Example: `"2015-11-03"`

iii. Value: text string of length [1..1000]

iv. If not supplied: `none`

d. The `"contestJurisdiction"` optional

i. Text description of the jurisdiction of this contest

ii. Example: `"Portland, ME"`

iii. Value: text string of length [1..1000]

iv. If not supplied: `none`

e. `"contestOffice"` optional

i. text description of the office being contested

ii. Example: `"Mayor"`

iii. Value: text string of length [1..1000]

iv. If not supplied: `none`

- `"cvrFileSources"` required List of input CVR file paths and their associated parameters. Multiple CVRs can be input in a single configuration file.

Each "cvrFileSources" list item contains the following parameters:

a. `"provider"` required

i. text description of the vendor / machine which generated this file

ii. value: `"cdf"` | `"clearBallot"` | `"dominion"` | `"ess"` | `"hart"`

b. `"filePath"` required

i. location of the CVR file (in the case of CDF, Clear Ballot, and ES&S), or the CVR folder (in the case of Dominion and Hart)

ii. example: `"/Users/test_data/2015-portland-mayor-cvr.xlsx"`

iii. value: string of length [1..255]

c. `"contestId"` required when CVR source files are from a provider other than ES&S must be blank for ES&S

i. the ID of the contest to tabulate, as represented in the CVR file(s)

ii. example: `"b651b997-417a-46d9-a676-a43d4df94ddc"`

iii. value: text string of length [1..1000]

d. `"firstVoteColumnIndex"` required and used if and only if the provider is ES&S

i. index of the column (starting from `1`) that contains the top-ranked candidate for each CVR

ii. example: `3`

iii. value: [1..1000]

e. `"firstVoteRowIndex"` required and used if and only if the provider is ES&S

i. index of the row (starting from `1`) that contains the rankings for the first CVR

ii. example: `2`

iii. value: [1..100000]

f. `"idColumnIndex"` optional and can be used only if the provider is ES&S

i. index of the column (starting from `1`) that contains the unique ID for each CVR

ii. example: `1`

iii. value: [1..1000]

g. `"precinctColumnIndex"` required and used if and only if `"tabulateByPrecinct"` is enabled and the provider is ES&S

i. index of the column (starting from `1`) that contains the precinct name for each CVR

ii. example: `2`

iii. value: [1..1000]

h. `"overvoteLabel"` optional and can be used only if the provider is ES&S or CDF

i. label used in the CVR to denote an overvote; if this parameter is present `"overvoteRule"` must be either `"alwaysSkipToNextRank"` or `"exhaustImmediately"` (because the other option, `"exhaustIfMultipleContinuing"`, relies on knowing which specific candidates were involved in each overvote)

ii. example: `"OVERVOTE"`

iii. value: string of length [1..1000]

i. `"undervoteLabel"` optional and can be used only if the provider is ES&S

i. the special label used in the cast vote records to denote an undervote

ii. example: `"UNDERVOTE"`

iii. value: string of length [1..1000]

j. `"undeclaredWriteInLabel"` optional

i. the special label used in the cast vote records to denote a vote for an undeclared write-in

ii. example: `"UWI"`

iii. value: string of length [1..1000]

k. `"treatBlankAsUndeclaredWriteIn"` optional and can be used only if the provider is ES&S

i. tabulator will interpret a blank cell in a CVR as a vote for an undeclared write-in

ii. value: `true` | `false`

iii. if not supplied: `false`

- `"candidates"` required List of registered candidate names and associated candidate code (note: leave empty when CVR is in Common Data Format)

  Each `"candidates"` list item has the following parameters:

a. `"name"` required

i. Full name of the registered candidate

ii. Example: `"Duson, Jill C."`

iii. Value: string of length [1..1000]

b. `"code"` optional

i. Candidate code which may appear in CVRs in lieu of full candidate name

ii. Example: `"JCD"`

iii. Value: string of length [1..1000]

iv. If not supplied: `none`

c. `"excluded"` optional

i. Candidate should be ignored during tabulation

ii. Value: `true` | `false`

iii. If not supplied: `false`

- `"rules"` required Set of configuration parameters that specify the tabulation rules The `"rules"` section contains the following parameters:

a. `"tiebreakMode"` required
   i. how the program should decide which candidate to eliminate when multiple candidates are tied for last place
   ii. or which candidate to elect first when:
   i. `"electionWinnerMode"` is set to `"multiWinnerAllowOnlyOneWinnerPerRound"`, and
   ii. multiple candidates exceed the winning threshold in the same round, and
   iii. at least two of those candidates are tied for the highest vote total in that round
   iii. value: `"random"` | `"stopCountingAndAsk"` | `"previousRoundCountsThenRandom"` | `"previousRoundCountsThenAsk"` | `"useCandidateOrder"` | `"generatePermutation"`
   i. we use `java.util.random` for randomness in our tiebreak implementations
   i. see: https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/Random.html
   ii. compatible methods exist for other languages, e.g. https://pypi.org/project/java-random/
   ii. on tabulation start a `java.util.Random` object is created if required using the randomSeed value specified in the input config file:
   i. `Random random = new Random(config.getRandomSeed());`
   iv. `"Random"`
   i. during tabulation, in the event of a tie at the end of a round:
   i. the list of tied candidates is sorted alphabetically
   ii. a `randomDouble` is generated using the random object:
   iii. `double randomDouble = random.nextDouble();`
   iv. the `randomDouble` is mapped to one of the tied candidates in the list:
   v. `int randomCandidateIndex = (int) Math.floor(randomDouble * (double) tiedCandidates.size());`
   vi. the selected candidate will be the winner or loser for that round
   v. `"stopCountingAndAsk"`:
   i. the user is presented with a list of tied candidates
   ii. the user will input their selection manually
   vi. `"previousRoundCountsThenRandom"`
   i. the tied candidate with the highest vote total in the previous round is selected
   ii. if there is a tie for the vote count in the previous round as well, a candidate is selected from the still tying candidates as described under `"tiebreakMode": "Random"`
   vii. `"previousRoundCountsThenAsk"`
   i. the tied candidate with the highest vote total in the previous round is selected
   ii. if there is a tie for the vote count in the previous round as well, a candidate is selected from the still tying candidates as described under `"tiebreakMode": "Stop counting and ask"`
   viii. `"useCandidateOrder"`
   i. during tabulation, in the event of a tie at the end of a round the list of candidates from the config file is consulted
   ii. if selecting a winner the tied candidate in this round who appears earliest is selected as a winner
   iii. if selecting a loser the tied candidate who appears latest is selected as the loser.
   ix. `"generatePermutation"`
   i. on config load candidate names are sorted alphabetically by candidate code, or if code is not present, candidate name
   ii. a randomly ordered candidate permutation is created using `Collections.shuffle()` with the `randomSeed` specified in the input config file
   iii. during tabulation, in the event of a tie at the end of a round, this permutation is consulted
   iv. if selecting a winner: the tied candidate in this round who appears earliest is selected
   v. if selecting a loser: the tied candidate who appears latest is selected

b. `"overvoteRule"` required

i. how the program should handle an overvote when it encounters one

ii. value: `"alwaysSkipToNextRank"` | `"exhaustImmediately"` | `"exhaustIfMultipleContinuing"`

iii. `"exhaustImmediately"`

i. `"exhaustImmediately"`: exhaust a ballot as soon as we encounter an overvote

c. `"winnerElectionMode"` required

i. which process the program should apply for selecting the winner(s)

ii. value: `"singleWinnerMajority"` | `"multiWinnerAllowOnlyOneWinnerPerRound"` | `"multiWinnerAllowMultipleWinnersPerRound"` | `"bottomsUp"` | `"bottomsUpUsingPercentageThreshold"` | `"multiPassIrv"`

iii. If only `singleWinnerMajority` is in use.

i. `"singleWinnerMajority"`: no special process (only valid when `numberOfWinners = 1`).

i. Election threshold = floor(V/(S+1)) + 1

ii. where `V` = total number of votes (in the current round); and `S` = numberOfWinners

d. `"numberOfWinners"` required

i. the number of seats to be won in this contest

ii. Uneditable if using `"single-winner majority determines winner"`, automatically set to `1`. Uneditable if using `"Bottoms-up using percentage threshold"`, automatically set to `0`.

iii. note: we use fractional vote transfer to redistribute votes in `"multiWinnerAllowOnlyOneWinnerPerRound"` and `"multiWinnerAllowMultipleWinnersPerRound"` contests.

iv. note: `0` is valid only when `"winnerElectionMode"` is set to `"bottomsUpUsingPercentageThreshold"`

v. value: [0..number of declared candidates]

e. `"minimumVoteThreshold"` optional

i. if a candidate receives fewer than this number of votes in the first round, they are automatically eliminated

ii. example: `150`

iii. value: [0..1000000]

iv. if not supplied: no automatic elimination occurs (equivalent to setting it to `0`)

v. Note: If no candidate exceeds the minimum vote threshold, tabulation silently fails. If you are using the minimum vote threshold setting and are having issues getting results, check that you have not set the minimum vote threshold too high.

f. `"maxSkippedRanksAllowed"` required

i. maximum number of skipped ranks (undervotes) on a ballot before the ballot should be considered exhausted; if `"unlimited"` is entered, a ballot will never be considered exhausted due to skipped ranks

ii. example: `1`

iii. value: [unlimited, 0..1000000]

g. `"maxRankingsAllowed"` required

i. maximum number of candidates that a ballot is allowed to rank; if `"max"` is entered, this will default to the total number of declared candidates as entered on the candidates tab

ii. example: `15`

iii. values: [max, 1..1000000]

h. `"rulesDescription"` optional

i. text description of this rules configuration for organizing your config files -- not used by the tabulator

ii. Example `"Maine Rules"`

iii. value: string of length [1..1000]

i. `"batchElimination"` optional

i. Tabulator will use batch elimination (only valid for single-winner contests)

ii. Value: `true` | `false`

iii. If not supplied: `false`

j. `"continueUntilTwoCandidatesRemain"` optional

i. tabulator will keep tabulating (beyond winning round) until only two candidates remain (only valid for single-winner contests)

ii. Value: `true` | `false`

iii. If not supplied: `false`

k. `"overvoteDelimiter"` optional and can be used only if provider is ES&S must be blank when `"overvoteLabel"` is provided

i. string that will be used to split a cell into multiple candidate strings in the case of an overvote

ii. example: `//`

iii. value: any string that contains no backslashes and at least one character that is not a letter, number, hyphen, period, comma, apostrophe, quote, or space

l. `"tabulateByPrecinct"` optional

i. Tabulator will generate a results spreadsheet for each precinct

ii. Value: `true` | `false`

iii. If not supplied: `false`

m. `"generateCdfJson"` optional

i. Tabulator will generate a JSON of cast vote records in the Common Data Format

ii. Value: `true` | `false`

iii. If not supplied: `false`

iv. `"exhaustOnDuplicateCandidate"` optional

i. Tabulator will exhaust a ballot when it encounters a duplicate candidate (instead of just skipping the duplicate)

ii. Value: `true` | `false`

iii. If not supplied: `false`

n. `"nonIntegerWinningThreshold"` optional

i. the vote threshold used to determine winners can be a non-integer

ii. if `true`, threshold = V/(S+1) + 10^-d

iii. if `false`, threshold = floor(V/(S+1)) + 1

iv. where `V` = total number of votes (in the current round or in the first round, depending on the winnerElectionMode); `S` = numberOfWinners; and d = decimalPlacesForVoteArithmetic

v. (note that `S+1` in the formulas above becomes just `S` if `"hareQuota"` is set to `true`.)

vi. Only valid for `"multiWinnerAllowOnlyOneWinnerPerRound"` and `"multiWinnerAllowMultipleWinnersPerRound"` contests

vii. value: `true` | `false`

viii. if not supplied: `false`

o. `"hareQuota"` * optional

i. the winning threshold should be computed using the Hare quota* (floor(votes divided by seats)) instead of the preferred Droop quota (votes divided by (seats+1))

ii. Only valid for `"multiWinnerAllowOnlyOneWinnerPerRound"` and `"multiWinnerAllowMultipleWinnersPerRound"` contests

iii. Value: `true` | `false`

iv. If not supplied: `false`

p. `"randomSeed"` required if `"tiebreakMode"` is `"random"`, `"previousRoundCountsThenRandom"`, or `"generatePermutation"`

i. the integer seed for the application's pseudorandom number generator

ii. value: [-140737488355328..140737488355327]

q. `"multiSeatBottomsUpPercentageThreshold"` required if `"winnerElectionMode"` is `"bottomsUpUsingPercentageThreshold"` and `"numberOfWinners"` is `0`

i. the percentage threshold used to determine when to stop the tabulation and declare winners

ii. note: only valid when `"winnerElectionMode"` is `bottomsUpUsingPercentageThreshold` and `"numberOfWinners"` is `0`

iii. value: [1..100]

r. `"decimalPlacesForVoteArithmetic"` required

i. number of rounding decimal places when computing winning thresholds and fractional vote transfers

ii. note: only editable when `"winnerElectionMode"` is `"multiWinnerAllowOnlyOneWinnerPerRound"` or `"multiWinnerAllowMultipleWinnersPerRound"`

iii. value: [1..20]

s. `"winnerElectionMode"` required

i. which process the program should apply for selecting the winner(s)

ii. value: `"singleWinnerMajority"` | `"multiWinnerAllowOnlyOneWinnerPerRound"` | `"multiWinnerAllowMultipleWinnersPerRound"` | `"bottomsUp"` | `"bottomsUpUsingPercentageThreshold"` | `"multiPassIrv"`

iii. `"multiWinnerAllowOnlyOneWinnerPerRound"`: elect no more than one winner per round, even when there are multiple candidates exceeding the winning threshold (only valid when `"numberOfWinners"` is `> 1`)

i. Election threshold = floor(V/(S+1)) + 1

ii. where V = total number of votes (in the first round); and S = numberOfWinners

iv. `"multiWinnerAllowMultipleWinnersPerRound"`: may elect more than one winner per round when there are multiple candidates exceeding the winning threshold (only valid when `"numberOfWinners"` is `> 1`)

i. Election threshold = floor(V/(S+1)) + 1

ii. where `V` = total number of votes (in the first round); and `S` = numberOfWinners

v. `"bottomsUp"`: instead of running a standard multi-seat contest with single transferable votes, just eliminate candidates until there are numberOfWinners remaining (only valid when `"numberOfWinners"` is `> 1`)

i. Election threshold = floor(V/(S+1)) + 1

ii. where `V` = total number of votes (in the first round); and `S` = `"numberOfWinners"`

iii. `"bottomsUp"` does not rely on election threshold for any vote processing

vi. `"bottomsUpUsingPercentageThreshold"`: instead of running a standard multi-seat contest with single transferable votes, just eliminate candidates until all remaining candidates have vote shares that meet or exceed `"multiSeatBottomsUpPercentageThreshold"` (only valid when `"numberOfWinners"` is `0`)

i. Election threshold = V•T

ii. where `V` = total number of votes (in the current round); and `T` = `"multiSeatBottomsUpPercentageThreshold"`

vii. `"multiPassIrv"`: instead of running a true multi-seat contest, run a series of single-seat contests and progressively exclude candidates as they win seats (only valid when `"numberOfWinners"` is `> 1`).

i. Election threshold = floor(V/(2)) + 1

ii. where `V` = total number of votes (in the current round)

t. `"overvoteRule"` required

i. how the program should handle an overvote when it encounters one

ii. value: `"alwaysSkipToNextRank"` | `"exhaustImmediately"` | `"exhaustIfMultipleContinuing"`

iii. `"alwaysSkipToNextRank"`: when we encounter an overvote, ignore this rank and look at the next rank in the cast vote record

iv. `"exhaustIfMultipleContinuing"`: if more than one candidate in an overvote are continuing, exhaust the ballot; if only one, assign the vote to them; if none, continue to the next rank (not valid with an ES&S source unless `"overvoteDelimiter"` is supplied)

2.31 Section 26 - RCTab CVR Files

This document describes the different CVR files RCTab is compatible with. It documents how they are laid out and describes relevant file structures that inform how RCTab parses CVR data.

2.31.1 ES&S

The CVR files from the ES&S EVS export system are in an MS Excel Worksheet format. The sample below displays the following records:

- **First Vote Column Index** – the first vote appears in column D under "Cand Choice 1"
- **First Vote Row Index** – the first cast vote record is in Row 2 in this example
- **ID Column Index** – in this sample Column A contains all Cast Vote Record IDs, making it the ID Column
- **Precinct Column Index** – displayed in Column B as the precinct Name/#
- **Ballot Style** – in Column C is the Name/# of the ballot style
- **Cand Choice** – in columns D through H are the candidate choices selected by the voter

	A	B	C	D	E	F	G	H
1	Cast Vote Record	Precinct	Ballot Style	Cand Choice 1	Cand Choice 2	Cand Choice 3	Cand Choice 4	Cand Choice 5
2	1	1	1	Harvey Curley	Larry Edwards	Mary Hall-Rayford	Sarah Lucido	Write-In
3	2	1	1	Larry Edwards	Mary Hall-Rayford	Sarah Lucido	Write-In	Harvey Curley
4	3	1	1	Mary Hall-Rayford	Sarah Lucido	Write-In	Harvey Curley	Larry Edwards
5	4	1	1	Sarah Lucido	Write-In	Harvey Curley	Larry Edwards	Mary Hall-Rayford
6	5	1	1	Write-In	Harvey Curley	Larry Edwards	Undervote	Undervote
7	6	1	1	Harvey Curley	Larry Edwards	Mary Hall-Rayford	Sarah Lucido	Write-In
8	7	1	1	Larry Edwards	Mary Hall-Rayford	Sarah Lucido	Undervote	Harvey Curley
9	8	1	1	Mary Hall-Rayford	Sarah Lucido	Write-In	Harvey Curley	Larry Edwards
10	9	1	1	Sarah Lucido	Write-In	Harvey Curley	Larry Edwards	Mary Hall-Rayford
11	10	1	1	Write-In	Harvey Curley	Larry Edwards	Mary Hall-Rayford	Sarah Lucido

Figure 1 - Sample cvr File

Note: The first three columns in the ES&S' file are fixed and the number of columns will depend on the number of contest choices. The format will typically follow this format:

Header Row (column labels)

Cast Vote Record	Precinct	Ballot Style	CONTEST-X 1st Choice	CONTEST-X 2nd Choice	CONTEST-X 3rd Choice	CONT 4th C

Each CVR Row

Column Number	Description
Column 1	Cast Vote Record ID number. Unique number, sequentially assigned.
Column 2	Precinct Name that is associated with the cast vote record.
Column 3	Ballot Style Name that is associated with the cast vote record. Applies to both ballots by precinct and ballots by style elections.
Column 4 thru N + 3	`N + 3` where `N` is the number of choices on the ballot. The content of the cell in each of the columns will be the name of the candidate selected on the ballot. • If there is no selection for the choice, the content of the column will show "undervote". • If there is more than one selection for the choice, the column will show "overvote". • If the choice is a write-in, the column will show "write-in".

2.31.2 Hart Verity

Hart's Verity system exports CVRs in a multi-folder format and differs from that of singular file CVRs such as ES&S. The structure of Verity's CVRs are as follows:

CVR Archives Naming Convention

The format of the archive name is:

`CVRArchive_<C>_<S>_<E>_<T>.zip` where:

- `<C>` is the number of CVRs contained in the Archive.
- `<S>` is the number of cast sheet 1s.
- `<E>` is the number of cast electronic ballots.
- `<T>` is a timestamp used to make the archive name unique.

Cryptographically signing Hart CVRs

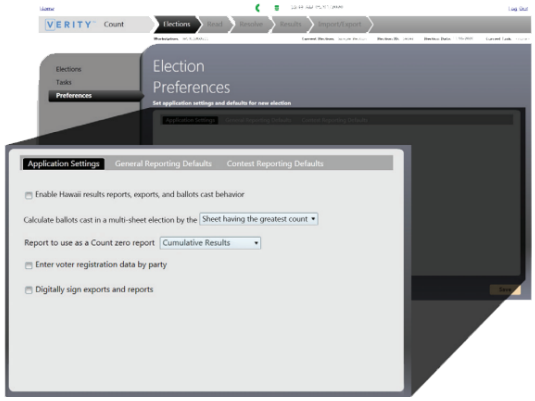
RCTab requires the Hart CVRs to be cryptographically signed. Signed CVR archives include a corresponding `.zip`.sig.xml file for the `.zip` as well as a corresponding `.xml.sig.xml` for each `.xml` CVR file inside the `.zip`. This allows the CSV `.zip` archive and its contents to be programmatically verified by RCTab during tabulation. Tabulation will not proceed without a `.sig.xml` for each CVR `.xml` file.

Use the Preferences menu in Verity Count to digitally sign CVR exports:

setting preferences

The Preferences menu allows you to set default preferences for all elections in Verity Count. You can set preferences at any time; however, with the exception of Application settings (the first tab), preferences set here only affect elections that have not yet been imported in Election Management. The first time you open Count you should set preferences before importing your first election onto the workstation. You can change settings for an election you have already imported on the Count workstation under the Results tab, in the Reporting Options menu (page 454).

1. Select the **Preferences** menu.
2. Select the **Application Settings** tab. Unlike other preferences, changing these settings will also apply to currently loaded elections.



A) *(Hawaii only)* Check the box to enable Hawaii-specific reports, exports, and ballots cast behavior.

B) *(Hawaii only)* If Hawaii ballots cast behavior is enabled, select how ballots cast are calculated in multi-sheet elections - either by the sheet having the highest count, or by the first sheet.

C) Choose which type of report you will be using as a basis for the Count Zero report, so that the application can validate that the Zero report has been printed.

D) Check the box if you would like to enter voter registration by party *(primary elections only)*.

E) Check the box if you would like to digitally sign all Count exports and reports.

- When a digital signature is generated, a separate file signature file is saved each time a report file or export is generated.
- A digital signature file may be used to verify that the report/export was produced using a Hart workstation.
- Note that all Audit Log, System Log, and Device Log reports are automatically signed. Checking this box will additionally sign all Count reports and exports.

CVR Structure

CVR data is recorded in XML file format, and can be viewed using a number of applications, such as Windows Notepad and Microsoft Edge. However, applications designed to work specifically with XML, such as XML Notepad, may simplify the presentation of the XML data.

XML files must be extracted from the ZIP file before being viewed. Once extracted, the "Open with" command may be used to view the XML in the desired application.

CVR File Naming Convention

The .zip contains many CVR files. Each CVR file name uses the convention `<S>_<G>.xml`, where:

- `<S>` is the CVR type identifier
- `1` if paper ballot sheet 1, blank for all other sheets
- `e` for DRE ballot
- `p` for Provisional ballot
- `<G>` is the GUID of the CVR

Examples: `1_c7db3cf2-c1a2-4773-971c-337ea2d198d7.xml`, or `50810795-95d4-4c0c-a6dc-0d5333e99c5a.xml`

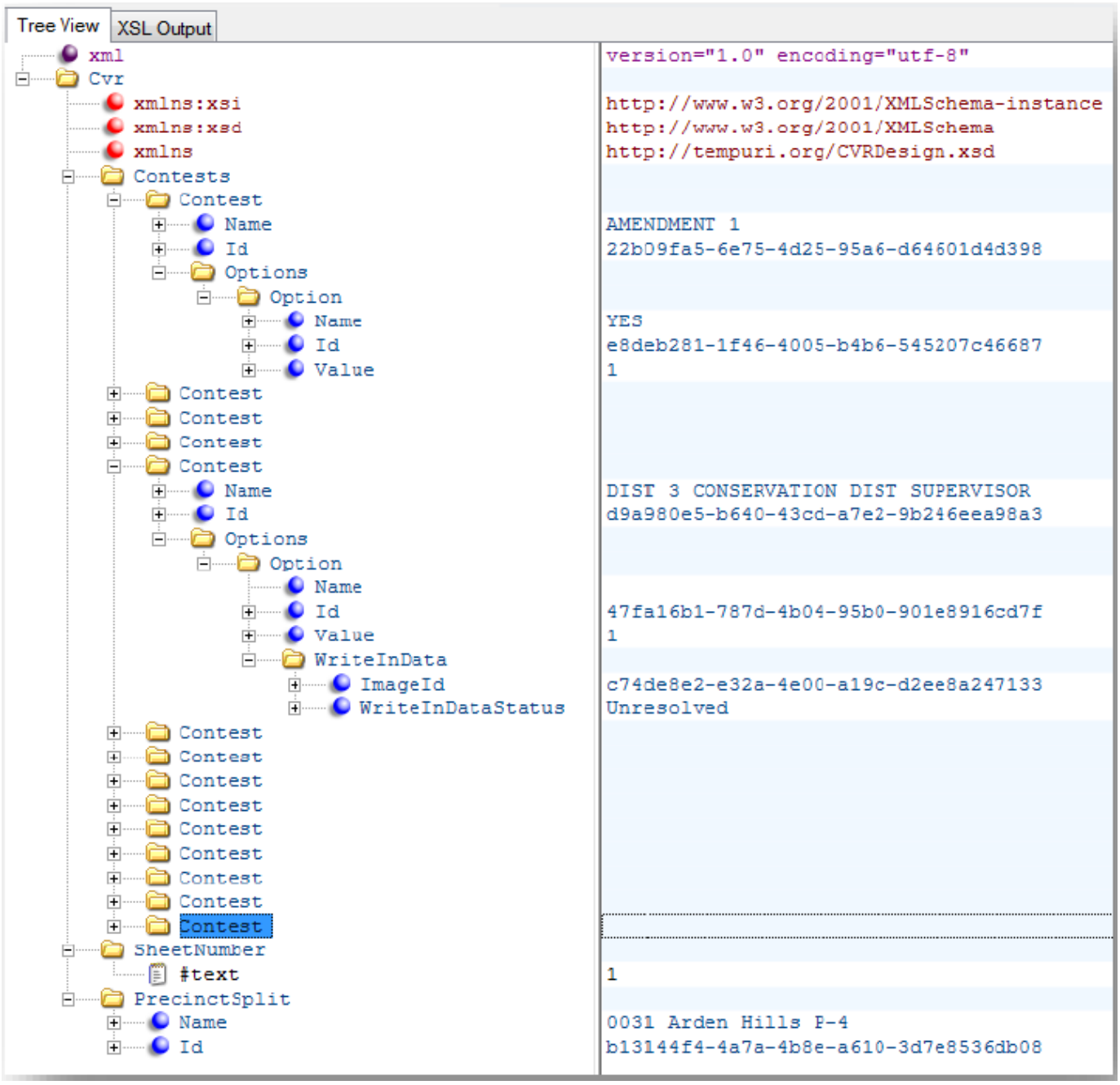
When signed, each `.xml` CVR file has a corresponding `.xml.sig.xml` file.

Viewing CVR Contents

The following is a general example of a CVR as viewed in Internet Explorer, showing two contests expanded:

```
<?xml version="1.0" encoding="UTF-8"?>
- <Cvr xmlns="http://tempuri.org/CVRDesign.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    - <Contests>
        - <Contest>
            <Name>AMENDMENT 1</Name>
            <Id>22b09fa5-6e75-4d25-95a6-d64601d4d398</Id>
          - <Options>
              - <Option>
                  <Name>YES</Name>
                  <Id>e8deb281-1f46-4005-b4b6-545207c46687</Id>
                  <Value>1</Value>
                </Option>
            </Options>
        </Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      - <Contest>
            <Name>DIST 3 CONSERVATION DIST SUPERVISOR </Name>
            <Id>d9a980e5-b640-43cd-a7e2-9b246eea98a3</Id>
          - <Options>
              - <Option>
                  <Name/>
                  <Id>47fa16b1-787d-4b04-95b0-901e8916cd7f</Id>
                  <Value>1</Value>
                - <WriteInData>
                    <ImageId>c74de8e2-e32a-4e00-a19c-d2ee8a247133</ImageId>
                    <WriteInDataStatus>Unresolved</WriteInDataStatus>
                  </WriteInData>
                </Option>
            </Options>
        </Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
      + <Contest>
    </Contests>
    <SheetNumber>1</SheetNumber>
  - <PrecinctSplit>
        <Name>0031 Arden Hills P-4</Name>
        <Id>b13144f4-4a7a-4b8e-a610-3d7e8536db08</Id>
    </PrecinctSplit>
</Cvr>
```

Here is the same section of the previous CVR as viewed in XML Notepad:



Structure

Each CVR accounts for all contests available to the voter for that ballot style (precinct-split/party/sheet combination), including a voter's marked choice(s), any remaining Undervotes, or for unmarked contests in an Open Primary, a "No vote" tag.

Marked choices are viewable in plain text. For marked write-ins, an ID is provided to cross-reference with a folder containing images of each write-in.

The following example combines all election types—General, Closed Primary, and Open Primary—into a "master" structure; not all fields will be present for all election types, and will be indicated appropriately.

CVR Elements

- `Cvr` (top-level folder)
- `Contests` (contains all contests for this ballot or sheet)
- `Contest` (random order of contests—Verity 1.0)
- `Name` (Title of contest; includes Party designation for Closed and Open Primaries)
- `Id` (Globally Unique ID [GUID] of contest)
- `Options` (contains each marked Choice for this contest)
- `Option` (empty if contest is completely undervoted)
- `Name` (Choice name, or blank if Write-in)
- `Id` (GUID of Choice)
- `Value` (`1` if marked; `100000` or similar, where the position of the 1 indicates a ranking mark in Ranked Choice contests, or to indicate one or more marks in a Cumulative contests)
- `WriteInData` (only present when Write-in Choice is marked)
- Text (Central vDrives only; text of Write-in, if resolved)
- `ImageID` (GUID of Write-in Image, and name of file located in "Write In Images" folder)
- `WriteInDataStatus` ("Unresolved" for all Scan write-ins; "Unresolved", "Resolved" or "Rejected" for all Central write-ins)
- `Undervotes` (quantity of undervotes)
- `Overvoted` (only present if contest is also overvoted)
- `NoVote` (Open Primaries only; indicates the contest was not voted in)
- `InvalidVote` (Open Primaries only; only present if contests of different party affiliations are voted in)
- `BatchSequence` (Central vDrives only; indicates the sheet's sequence within the batch number, below)
- `IsElectronic` (True/False; indicates whether the vote was cast with a DRE)
- `SheetNumber` (sheet indicator of paper ballot)
- `PrecinctSplit`
- `Name` (name of the voted Precinct-Split)
- `Id` (GUID of Precinct-Split)
- `Party` (Closed Primaries only; indicates party affiliation of ballot)
- `Name` (name of Party)
- `Id` (GUID of Party)
- `BatchNumber` (Central vDrives only; indicates the Batch ID)
- `DeviceSerialNumber` (Verity 1.0: Central vDrives only; indicates the vDrive ID)
- `RetrievalCode` (Unique Retrievable Ballot Code for this CVR, if applicable)
- `ProvisionalCode` (Unique Provisional Ballot Code for this CVR, if applicable)
- `CvrGuid` (GUID of CVR)

2.31.3 Dominion

Dominion's CVR format consists of several files which are grouped together in the same directory, similar to Hart's CVR structure. All exported files have a top level attribute which is set to the version of EMS that produced the file.

Configuration

This file shows which parameters were selected by the user when performing the export. It contains the following attributes:

- `IncludeTabulatorFilter` : Boolean value indicating whether the export was performed for a single tabulator.
- `TabulatorFilterValue` : a numeric identifier of the tabulator that was exported.
- `IncludeResultContainerFilter` : Boolean value indicating whether the batch filter was used.
- `ResultContainerFilterValue` : an integer indicating the batch to export CVR data for.
- `IncludePrecinctPortionFilter` : Boolean value indicating whether the export was performed for a single precinct portion.
- `PrecinctPortionFilterValue` : an integer indicating the precinct portion to export CVR data for.
- `IncludeBallotTypeFilter` : Boolean value indicating whether the export was performed for a single ballot type.
- `BallotTypeFilterValue` : an integer indicating the ballot type to export CVR data for.
- `IncludeContestFilter` : Boolean value indicating whether the export was performed for a single contest.
- `ContestFilterValue` : an integer indicating the contest to export CVR data for.
- `SplitFilesPerBatch` : a Boolean value indicating whether separate CVR export JSON files should be created per batch.
- `IncludeOnlyPublishedResultContainers` : a Boolean value indicating whether only published result containers where included.

`ContestManifest`

This file contains a list of all non-disabled contests ordered by global order that can produce votes. The list is ordered by contest global order and each contest has the following attributes:

- Description: name of the contest.
- `Id` : identifier of the contest (internal machine id)
- External Id: external identifier (optional)
- `VoteFor` : the number of votes allowed/number of positions to be elected.
- `NumOfRanks` : the number of rankings allowed to be made.

`CandidateManifest`

This file contains a list of all non-disabled candidates ordered by global order (first by contest, then by choice) that can produce votes (for example "No Candidate" are not included).

- Description: name of the candidate.
- `Id` : identifier of the candidate (internal machine id)
- External Id: external identifier (optional)
- `ContestId` : identifier of the contest this choice belongs to.
- `Type` : candidate type. `Regular` , `Writein` , `NoPreference` , `QualifiedWriteIn` .

`PartyManifest`

This file contains a list of political parties ordered by global order, each with the following attributes:

- Description: name of the political party
- `Id` : identifier of the party (internal machine id)
- External Id: external identifier (optional)

`PrecinctPortionManifest`

This file contains a list of all precinct portions ordered by global order, each with the following attributes:

- Description: name of the precinct portion
- `Id` : identifier of the precinct portion (internal machine id)
- External Id: external identifier (optional)

`BallotTypeManifest`

This file contains a list of all ballot types ordered by global order, each with the following attributes:

- Description: name of the ballot type
- `Id` : identifier of the ballot type (internal machine id)
- External Id: external identifier (optional).

`BallotTypeContestManifest`

This file contains information on which contests are used on which ballot types. Each relationship has the following attributes:

- `BallotTypeId` : identifier of the ballot type (internal machine id)
- `ContestId` : identifier of the contest (internal machine id).

`CountingGroupManifest`

This file contains a list of all counting groups ordered by global order, each with the following attributes:

- Description: name of the counting group
- `Id` : identifier of the counting group (internal machine id)
- External Id: external identifier (optional).

`TabulatorManifest`

This file contains a list of all tabulators ordered by global order, each with the following attributes:

- Description: name of the tabulator
- `Id` : identifier of the tabulator, we will use tabulator number here.
- `ExternalId` : the external string identifier (optional).
- `ThresholdMin` : minimum threshold for voting box scanned on this tabulator
- `ThresholdMax` : maximum threshold for voting box scanned on this tabulator
- `WriteinThresholdMin` : minimum threshold for write-in area scanned on this tabulator
- `WriteinThresholdMax` : maximum threshold for write-in area scanned on this tabulator

`CVRExport`

This file contains the actual CVR data. In addition to the top level version field it also has an ElectionId field that contains the description of the election.

The main content of the file is a list of CVR sessions. Each **Session** object contains the following attributes:

- `TabulatorId` : the tabulator id, same as the one used in the manifest
- `BatchId` : the batch id, unique for a given tabulator id.
- `RecordId` : the CVR id within the batch.
- `CountingGroupId` : the counting group id, same as the one used in the manifest.
- `ImageMask` : the file mask for finding the associated images with this session.
- Original element, contains the original state of the CVR data for this session.
- Modified element (optional), contains the modified state of the CVR data for this session.

Original/Modified element:

This element contains attributes that can be potentially be modified during adjudication/conditional voting management:

- `PrecinctPortionId` : the precinct portion id, same as the one used in the manifest.
- `BallotTypeId` : the ballot type id, same as the one used in the manifest.
- `IsCurrent` : set to true, if this element represents the current state of the CVR.
- Contest elements. Lists all contests for the current ballot type.

Contest:

This element represents a marked contest. Contains the following attributes:

- `Id` : contest identifier, as used in the manifest file.
- `Marks` : list of marked (explicitly/implicitly) in this contest. Note: explicit marks mean when the voter filled in the voting box directly. Implicit means when the voting box was implied by a straight party ticket selection.

Mark element

Contains the following attributes:

- `CandidateId` : indicates the candidate the mark is for (if a write-in position is resolved to a qualified write-in the candidate id will point to a qualified write-in).
- `PartyId` : indicates party affiliation. If not party affiliation then this will be `0` .
- `Rank` : indicates rank; will be `1` by default, will only contain values higher than `1` if ranked choice voting is used.
- `WriteinIndex` : if mark is for write-in position (or qualified write-in) this attribute indicates which write-in position in the contest (`0` means first, `1` means second position, etc.)
- `MarkDensity` : percentage that voting box was filled.
- `WriteinDensity` : percentage that write-in area was filled in. Attribute exists only if it is a write-in position.
- `IsAmbiguous` : a Boolean value indicating whether mark is ambiguous.
- `IsVote` : a Boolean value indicating whether the mark produced a vote Note: an implicit selection because of straight party vote would also be set to true. Any mark above the max threshold will be true in a ranked choice voting contest.

2.31.4 Clear Ballot

Clear Ballot's CVR is still currently under development and while RCTab was developed to include Clear Ballot as a vendor, it is considered a beta feature at this time. Once Clear Ballot has finalized a format for their CVR export, the details of said CVR will be included in this document as a reference.

2.32 Section 27 - RCTab Config Files

RCTab's Config File is generated by the software when creating a new election. The Config file is saved in a JSON format and should only be revised using RCTab software (do not attempt to edit the JSON file). The Config File contains all the information necessary to conduct an election. The file must be validated using RCTab prior to tabulating an election. See **Section 18 - User Guide** for more. Here is a sample of the Config File:


```
{
    "tabulatorVersion": "1.2.0",
    "outputSettings": {
        "contestName": "Test Contest",
        "outputDirectory": "C:\\Users\\ChrisHughes\\Desktop\\L&A Results",
        "contestDate": "2015-11-03",
        "contestJurisdiction": "Date",
        "contestOffice": "Mayor",
        "tabulateByPrecinct": false,
        "generateCdfJson": false
    },
    "cvrFileSources": [
        {
            "filePath": "C:\\Users\\ChrisHughes\\Desktop\\universal_rcv_tabulator_v1",
            "contestId": "",
            "firstVoteColumnIndex": "4",
            "firstVoteRowIndex": "2",
            "idColumnIndex": "1",
            "precinctColumnIndex": "2",
            "overvoteDelimiter": "",
            "provider": "ess",
            "overvoteLabel": "overvote",
            "undervoteLabel": "undervote",
            "undeclaredWriteInLabel": "UWI",
            "treatBlankAsUndeclaredWriteIn": false
        }
    ],
    "candidates": [
        {
            "name": "Brennan, Michael F.",
            "code": "",
            "excluded": false
        },
        {
            "name": "Bragdon, Charles E.",
            "code": "",
            "excluded": false
        },
        {
            "name": "Rathband, Jed",
            "code": "",
            "excluded": false
        },
        {
            "name": "Strimling, Ethan K.",
            "code": "",
            "excluded": false
        },
        {
            "name": "Vail, Christopher L.",
            "code": "",
            "excluded": false
        }
    ],
    "rules": {
        "tiebreakMode": "stopCountingAndAsk",
        "overvoteRule": "exhaustImmediately",
        "winnerElectionMode": "singleWinnerMajority",
        "randomSeed": "",
        "numberOfWinners": "1",
        "multiSeatBottomsUpPercentageThreshold": "",
        "decimalPlacesForVoteArithmetic": "4",
        "minimumVoteThreshold": "0",
        "maxSkippedRanksAllowed": "unlimited",
        "maxRankingsAllowed": "15",
        "nonIntegerWinningThreshold": false,
        "hareQuota": false,
        "batchElimination": true,
        "continueUntilTwoCandidatesRemain": true,
        "exhaustOnDuplicateCandidate": false,
        "rulesDescription": "Maine Rules",
        "treatBlankAsUndeclaredWriteIn": false
    }
}
```

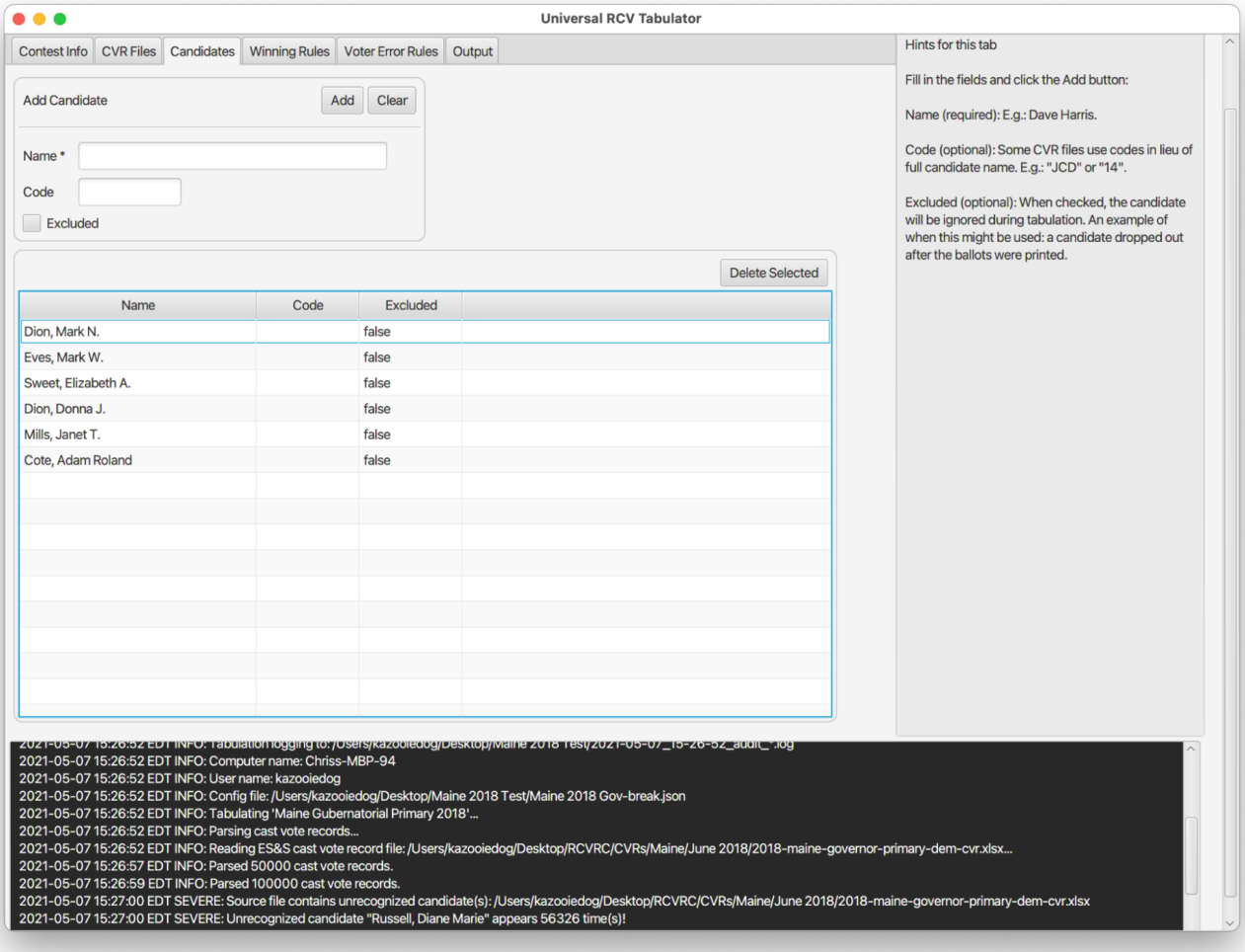
2.33 Section 28 - Post-Election Audit & Clearing RCTab from System

2.33.1 Post-Election Audit Preparation for RCTab

In order to run a post-election audit of RCTab, users will need to gather up all relevant log files and understand the information included in those logs. This guide will describe where to find the relevant log files and how to read those log files in any audit procedure.

Retrieving the Operator log(s)

The operator log is a `.log` file that includes all information from the black log box at the bottom of the user interface, as shown at the bottom of the screenshot here:



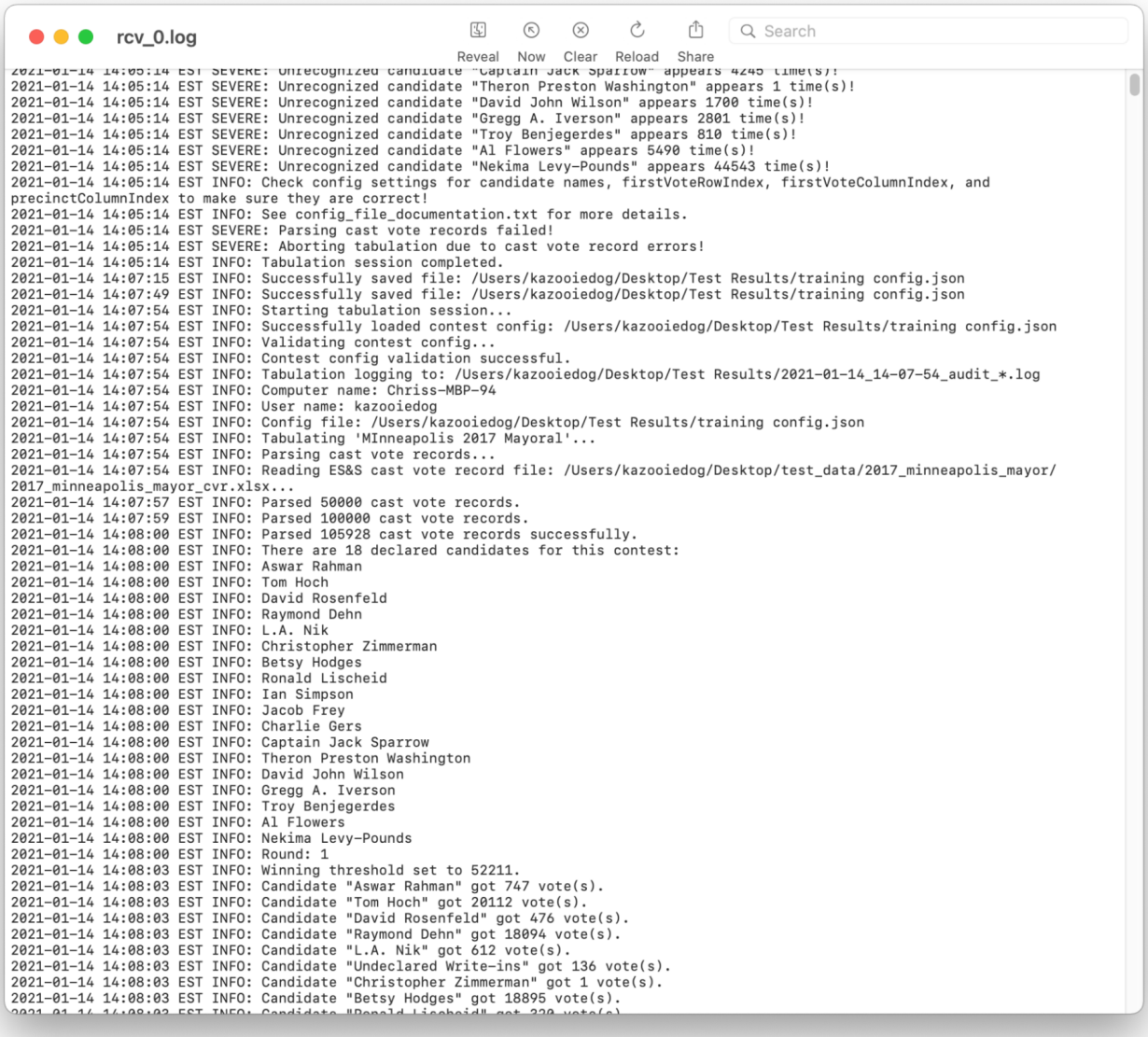
The operator log is updated any time RCTab generates any message to send through that log box. Note that messages are generated internally by the software, then saved to the operator log file and shown to the user in the log box.

On Windows, this log file is saved to the folder where RCTab is launched from - in practice, that means the bin folder within RCTab's install location (the bin folder is where the user launches RCTab using the rcv.bat launcher file). Operator logs are saved in the format `rcv_*.log`: `rcv_0.log`, `rcv_1.log`, and so on. Each log has a maximum size of 50MB, and once a log reaches 50MB, a new `rcv_*.log` file is created. This log file is continuously updated as a user interacts with RCTab. Users should retrieve all such logs when conducting a post-election audit.

Reading the operator log

The operator log includes any error messages sent to the user in operation, as well as summary information about each tabulation run through RCTab (how many CVRs have been read in, how many votes each candidate got in each round, winners of each contest). Below is a screenshot of a `rcv_0.log`. Each line starts with a timestamp of when the message was generated. There are a variety of messages in this screenshot: informational messages letting the user know they've saved a configuration file, information about a tabulated contest (how many votes each candidate received in a round, the candidate names RCTab is looking for in the CVR), and error messages about information RCTab needs in order to run a ranked choice voting election (at the top of the screen). Similar information will be repeated in the operator log file every time a user runs a tabulation through RCTab.

This log can be used to determine how a user has interacted with RCTab: any error messages they've received, summary information of any time they've run a tabulation on the given installation of RCTab, and other information about user interaction with the software itself.



All messages RCTab may send to a user are included in **Section 29 - RCTab Operator Log Messages**. This document categorizes messages by the labels given to them in RCTab's operation: "INFO", "WARNING", and "SEVERE". Severe errors cause RCTab's tabulation to fail, and so suggested resolution steps for any severe errors are provided in the document. "INFO" and "WARNING" messages convey information about contest configuration files, contest tabulation progress, CVR files, and other

relevant information about tabulation. No "INFO" or "WARNING" messages cause tabulation to fail, but "INFO" and "WARNING" messages may be sent along with Severe messages to provide users with potential resolution steps for any Severe errors.

Retrieving contest audit log(s)

The audit log is a record of everything RCTab did to process a given ranked choice voting contest. A new audit log is produced each time the user runs a tabulation using the "Tabulate" option in RCTab. Each audit log is saved to the output folder the user selects in the Output Directory setting on the Output tab in the RCTab user interface. The manufacturer suggests users save all files for a contest to a folder named after that contest (for example, County Commission November 2020, City Council District 5 April 2018). This will make retrieving any audit logs straightforward.

Audit logs, like operator logs, have a maximum size of 50MB. Once an audit log reaches 50MB, RCTab creates a new audit log for the tabulation. Audit log files grow quickly because they include a large amount of information on how every piece of data in a CVR file is handled, so users should confirm that they have retrieved every audit `.log` file from the relevant contest(s).

Audit logs are named according to the rule:

`<time_stamp>_audit_0.log`

where the timestamp is created when the tabulation is triggered and used on all audit files for a given tabulation.

For example, a tabulation that begins at 10:49:49 pm on April 24, 2021, will produce an audit log named:

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

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`

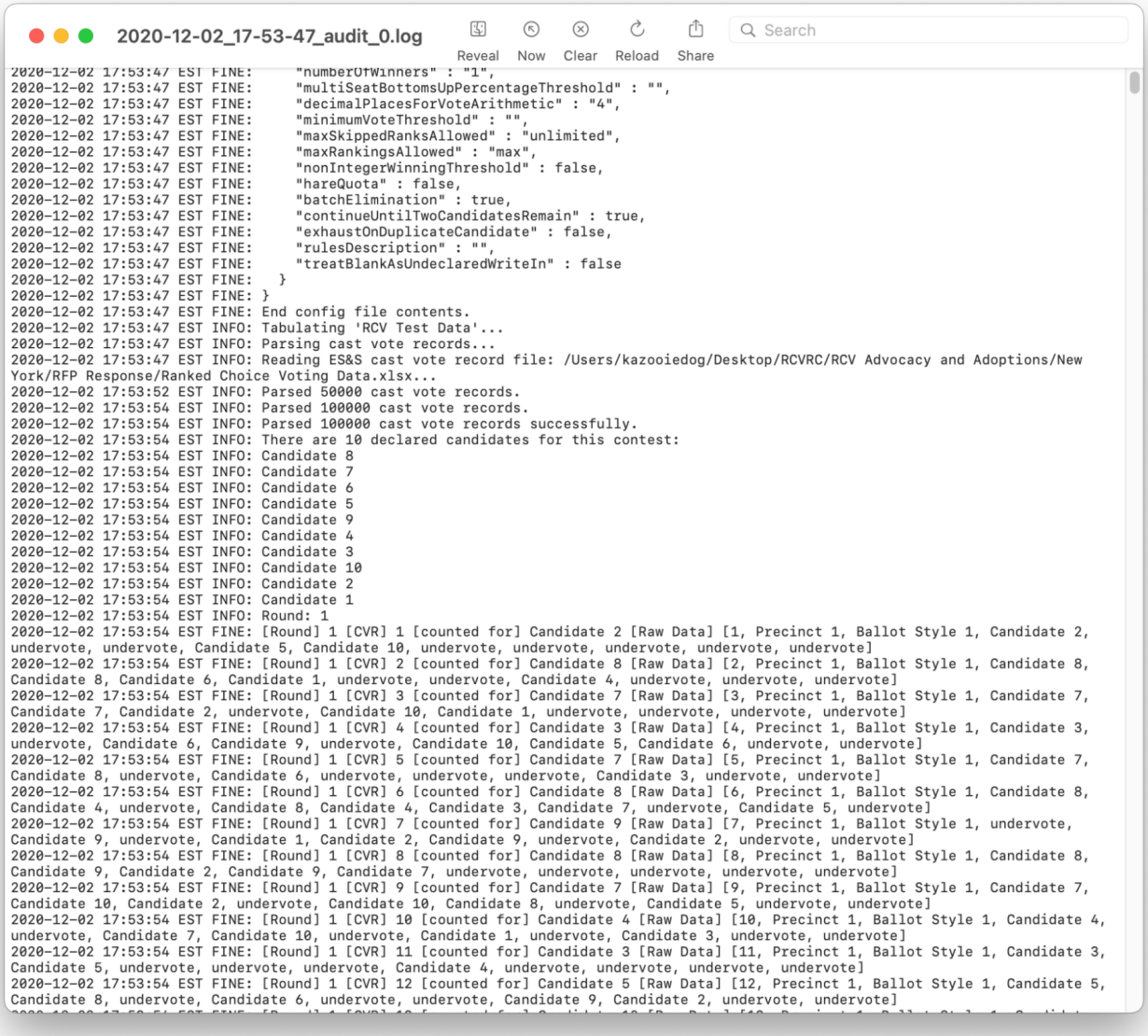
`2021-04-24_22-49-49_audit_1.log` -> `2021-04-24_22-49-49_audit_2.log`

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

Each time an audit log file is written to disk a corresponding .hash file is also written. This hash file contains a cryptographic hash that can be used to verify the contents on the corresponding audit log file. See **Section 23 - Trusted Build & Output Hash Verification - Windows OS** for instructions on how to use the hash file to verify audit logs.

Reading the audit log

Audit logs include the configuration file settings used in the tabulation, how every individual ballot/CVR record in the given CVR file(s) for the contest got counted in each round, which candidate got eliminated in each round of counting, and other details of how the contest was counted. These logs also include all messages sent in the log box on the Tabulator UI during contest tabulation. Below is a screenshot example of information included in an audit log:



Each line starts with a timestamp of when the message was generated.

At the top of the screenshot is the end of the configuration file settings for this contest. In the middle, RCTab is reading in all the CVR files for the contest and a list of all the candidate names listed in the configuration file that is also found in the CVR files. At the bottom is the start of ballot-by-ballot counting. In the first round of counting, every individual ballot in the CVR file(s) in a contest will be listed out here. This information displays how RCTab read in each ballot and which candidate it counted the ballot for in the first round. In all subsequent rounds, the audit log displays information for each transferred ballot - which candidate the ballot counted for in the previous round and which candidate the ballot counts for in that new round (or the exhaust condition of the ballot - if an overvote is reached or if a ballot runs out of rankings, for example).

Audit logs close by noting where summary results files were saved to in the case of a successful tabulation. In the case of a tabulation that cannot be completed (because candidate names are missing, or required rules are missing, or other issues), audit logs close by noting the errors sent to a user. This same information will be included in the operator log.

Audit log files can be used to check:

- What counting rules were used in a tabulation using RCTab;
- The list of candidates used in a tabulation using RCTab;
- The total number of individual cast vote records included in all CVR export files used in a contest tabulation;
- How every ballot in each CVR file was counted in each round of counting;
- Summary information for votes each candidate received in each round of the contest;
- How RCTab read every single cell of each CVR file;
- Where all contest files were saved;
- And errors leading to a failed tabulation.

This is how every line in a CVR file is first displayed in RCTab:

```
2020-12-02 17:53:54 EST FINE: \[Round\] 1 \[CVR\] 4582 \[counted for\] Candidate 9 \[Raw Data\] \[4582, Precinct 5, Ballot Style 5, Candidate 9, Candidate 4, undervote, undervote, Candidate
10, undervote, undervote, overvote, undervote, Candidate 2\]

This can be broken down into three parts: the time stamp, how the vote was counted in this round, and all the data RCTab read for this individual cast vote record.

Time Stamp:
2020-12-02 17:53:54 EST FINE: This message was sent at 5:53 pm on December 2, 2020.

Vote counted in this round:
\[Round\] 1 \[CVR\] 4582 \[counted for\] Candidate 9: In round 1, CVR ID 4582 was counted for Candidate 9.

All CVR data read for this individual CVR:
\[Raw Data\] \[4582, Precinct 5, Ballot Style 5, Candidate 9, Candidate 4, undervote, undervote, Candidate 10, undervote, undervote, overvote, undervote, Candidate 2\]: This ballot was
labeled CVR # 4582 in the CVR file for this contest. It was cast in precinct 5. The ballot style was ballot style 5. The ballot used their rankings in this order: Candidate 9, Candidate 4,
undervote, undervote, Candidate 10, undervote, undervote, overvote, undervote, Candidate 2.
```

(Note: undervote means a ranking was left blank).

Later in this same contest, after candidate 9 is eliminated, the audit log displays this information:

```
2020-12-02 17:53:56 EST FINE: \[Round\] 4 \[CVR\] 4582 \[transferred to\] Candidate 4
```

This tells the user that the CVR listed above, #4582, transferred in Round 4 to Candidate 4.

Users can check multiple different factors using the audit log and operator log to review the performance of RCTab software.

1. Using the audit `.log`, check that the total CVRs read into RCTab is equal to the total ballots cast according to the EMS used to export CVR files.
2. Check the `rcv_0.log` and any audit `.log`s for **SEVERE** errors and to see how users resolved those errors.
3. Compare summary result information in the audit file(s) and the summary results files for a contest run through RCTab.
4. Review the election data form used for a contest (see **Section 11 - L&A**). Compare the information in this form to the configuration file used in a contest as reflected in the audit `.log` for that contest to confirm that the proper counting rules were used for the contest.
5. Using audit `.log` files, compare how data in the CVR files was read into RCTab with the information included in CVR files exported from the relevant election management software. This data can also be cross-referenced with paper ballot data from precincts, using precinct information in the audit `.log` files and in CVR files from the election management software.
6. If using the Tabulate by Precinct feature of RCTab, users can compare a hand count of ballots from a precinct to RCTab's tabulation of those same ballots. The Tabulate by Precinct feature produces round-by-round results at the precinct level. These results display how ballots at the precinct level transferred in the contest as a whole, not a simulated round-by-round count in the precinct. A hand count could be conducted at the precinct level, following the elimination order in RCTab results, to check that RCTab counted each ballot properly in a given precinct.

a. Note that the Tabulate by Precinct feature identifies a winner. That identified winner is the winner of the contest overall, not necessarily the person who received the most votes in that specific precinct. This will be updated in a future release to identify the winner of the precinct.

7. When summary files are written to disk the audit log also contains the text of the hash of those files. Confirming that the summary file hashes in the audit log matches the .hash files written to disk is another layer of security to prevent malicious editing of summary files.

2.33.2 Clearing RCTab from a System

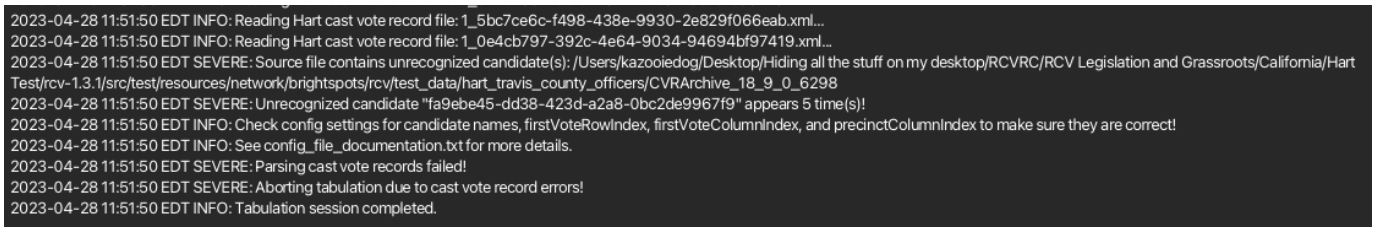
Note: Before following this procedure, ensure that any materials that must be archived according to the jurisdiction's archiving requirements have been archived.

There are three types of files needed to remove RCTab from your system: RCTab installation folders, any files you created when using RCTab (configuration files, results files, audit files), and any CVR files you save to the RCTab workstation.

- Delete the RCTab installation folder (generically called rctab_1.3.0_windows after you unzip it) - this deletes RCTab itself. Delete any .zip file of RCTab as well.
- Delete any folders where you have RCTab files (meaning configuration files, results files, and audit files) or CVR files saved. The simplest way to keep track of this is to set up an RCTab Files folder on the computer with RCTab, with subfolders for each contest/L&A process run through RCTab. The user could then save all relevant files for each contest (configuration file, results files, audit files, and CVR files) to those folders.

2.34 Section 29 - RCTab Operator Log Messages

RCTab generates messages and displays them in the Operator Log at the bottom of the user interface window. The log will list "SEVERE" messages, "WARNING" messages, and "INFO" messages for the operator to review and take appropriate action.



On the following pages are all the "SEVERE", "WARNING", and "INFO" messages that RCTab may log during the setup and tabulation of an RCV election. Each "SEVERE" error includes a suggested resolution step or steps directly below the relevant error. Some "SEVERE" errors can all be resolved with the same resolution step and are grouped together before the suggested resolution.

Errors in the operator log will include detailed information about errors encountered. Users should rely on that information, as well as information provided in this guide, to resolve errors encountered in tabulation. If following those steps still does not resolve your problem, contact the manufacturer for support. "WARNING" and "INFO" messages do not interrupt the ability of RCTab to complete tabulation and so do not have resolution steps suggested.

2.34.1 SEVERE MESSAGES

Configuration File Errors

Errors on this tab are correctable through the Tabulation drop-down menu on RCTab, by creating a new configuration file from scratch, or may require reference to other sections of this document.

```
SEVERE, "No contest config path specified!"
SEVERE, "Failed to load contest config: %s"
```

Suggested resolution: Load a configuration file using the "File" menu or create a new configuration file using the user interface and following **[Section 18 - User Guide]**(user_guide.md) document.

```
SEVERE, "Failed to load config."
```

Suggested resolution: Create new configuration using the user interface.

```
SEVERE, "Error logging config file: %s\n"
```

Suggested resolution: Create new configuration using the user interface.

```
SEVERE, "Invalid winnerElectionMode!"
SEVERE, "maxRankingsAllowed must either be \"%s\" or an integer from %d to %d!"
SEVERE, "maxSkippedRanksAllowed must either be \"%s\" or an integer from %d to %d!"
SEVERE, "tabulatorVersion %s not supported"
SEVERE, "tabulatorVersion must be set to %s"
SEVERE, "tabulatorVersion is required"
SEVERE, "Invalid tiebreakMode!"
SEVERE, "Invalid overvoteRule!"
SEVERE, "Invalid contestDate: %s!"
SEVERE, "Error parsing JSON file: %s\n%s\n" + "Check file formatting and values and make sure they are correct!\n"
                + "It might help to try surrounding values causing problems with quotes (e.g. \"value\").\n"
                + "See config_file_documentation.txt for more details."
SEVERE: "continueUntilTwoCandidatesRemain can't be true in a multi-seat contest unless the winner election mode is multi-pass IRV!"
SEVERE: "batchElimination can't be true in a multi-seat contest unless the winner election mode is multi-pass IRV!"
SEVERE, "winnerElectionMode can't be multiSeatSequentialWinnerTakesAll in a single-seat " + "contest!"
SEVERE, "winnerElectionMode can't be multiSeatBottomsUp in a single-seat contest!"
SEVERE, "winnerElectionMode can't be multiSeatAllowOnlyOneWinnerPerRound in a single-seat " + "contest!"
SEVERE, "batchElimination can't be true when winnerElectionMode is multiSeatBottomsUp!"
SEVERE, "%s should not be defined for CVR source with provider \"%s\" for file source %s", fieldName, provider, inputLocation
SEVERE, "treatBlankAsUndeclaredWriteIn should not be true for CVR source with provider \"%s\"for file source %s", fieldName, provider, inputLocation
SEVERE, "hareQuota can't be true when winnerElectionMode is \"%s\"!", winnerMode
```

```
SEVERE, "nonIntegerWinningThreshold and hareQuota can't both be true at the same time!"
SEVERE, "Failed to get tiebreaker!\n%s"
```

Suggested resolution: Create a configuration file using RCTab user interface. It should not be possible to select inconsistent sets of rules (which these errors are responding to) if a configuration is created using the UI. Confirm that the configuration file follows requirements set out in **Section 18 - User Guide** document.

```
SEVERE, "\"%s\" can't be used as %s if it's also being used as %s!" (%s = string, field, otherField)
```

Suggested resolution: Change one of these values so that different values are used for different options.

```
SEVERE, "%s must be an integer equal to %d for file source %s", fieldName, lowerBoundary, inputLocation
SEVERE, "%s must be an integer from %d to %d for file source %s", fieldName, lowerBoundary, upperBoundary, inputLocation
SEVERE, "%s must be an integer equal to %d if supplied for file source %s", fieldName, lowerBoundary, inputLocation
SEVERE, "%s must be an integer from %d to %d if supplied for file source %s", fieldName, lowerBoundary, upperBoundary, inputLocation
```

Suggested resolution: These errors appear when a number the user input into a number-based field (such as random seed or allowable number of consecutive skipped rankings) is outside the bounds usable by RCTab. Refer to the specific error message in the tabulator log, which will provide a range of numbers RCTab is compatible with for that specific field.

```
SEVERE, "Contest config validation failed! Please modify the contest config file and try again.\nSee config_file_documentation.txt for more details."
```

Suggested resolution: This error message will be preceded by specific error messages about information that RCTab finds missing in the configuration file.

```
SEVERE, "\"%s\" is a reserved term and can't be used for %s!"
```

Suggested resolution: Change the term used in the option setting identified in the error code. Re-save configuration file.

```
SEVERE, "Error during validation:\n%s\nValidation failed!"
```

Suggested resolution: Check operator error log for other error messages. Other error messages will appear explaining why validation could not be completed.

```
SEVERE, "Unable to process a config file with version %s using older version %s of the app!"
```

Suggested resolution: Create new configuration file using the user interface.

Contest Info Error

The error in this section is correctable on the Contest Info tab.

```
SEVERE, "contestName is required!"
```

Suggested resolution: Enter the contest name in the "Contest Info/Contest Name" field.

CVR Errors

Errors in this section are correctable on the CVR Files tab. They may also require detailed examination of CVR Files themselves and consultation with voting system vendor documentation to identify the source of any issues with CVR Files.

```
SEVERE, "filePath is required for each cast vote record file!"
```

Suggested resolution: Verify that a valid file location is entered in the "CVR Files/Path" field for each CVR included in the tabulation.

```
SEVERE, "No cast vote records found!"
```

Suggested resolution: Check that the folder or file RCTab is directed to has CVR files compatible with the RCTab software.

```
SEVERE, "Contest config must contain at least 1 cast vote record file!"
```

Suggested resolution: Select, in the "CVR Files" tab, at least one cast vote record file to tabulate.

```
SEVERE, "Cast vote record file not found: %s"
```

Suggested resolution: Confirm that the file path supplied for the CVR file listed in the error goes to a file accessible by RCTab.

```
SEVERE, "Duplicate cast vote record filePaths are not allowed: %s"
```

Suggested resolution: Delete all but one listing of each CVR File in the CVR list. CVR files should only be listed once in the CVR list.

```
SEVERE, "Error opening cast vote record file: %s"
```

Suggested resolution: Ensure that you have added a cast vote record file in a format recognized by RCTab.

```
SEVERE, "Aborting tabulation due to cast vote record errors!"
```

Suggested resolution: Review the log for other errors that will alert you to specific errors with your CVR files. Review your cast vote record files. Ensure that all candidate codes and other vote data are properly represented in configuration files.

```
SEVERE, "Parsing cast vote records failed!"
SEVERE, "Error parsing source file %s"
```

Suggested resolution: Check for other error messages in the operator log box. Parsing files fails when data appears in those files that the configuration file does not include. If information appears in the CVR files that is not in a configuration file, RCTab will abort tabulation. Ensure that CVR files are in a format RCTab can read.

```
SEVERE, "Data format error while parsing source file: %s"
```

Suggested resolution: Check that your CVR file is in the correct format. User may need to retrieve new CVR files.

```
SEVERE, "undeclaredWriteInLabel must be supplied if treatBlankAsUndeclaredWriteIn is true!"
```

Suggested resolution: Confirm that Undeclared Write-In Label is supplied on CVR tab.

```
SEVERE, "Invalid cell address: %s"
```

Suggested resolution: Check CVR files tab. Confirm that correct information explaining where information is included in CVRs is included for each CVR file.

```
SEVERE, "precinctColumnIndex is required when tabulateByPrecinct is enabled: %s"
```

Suggested resolution: Update configuration to include Precinct Column Index information for the CVR being processed. Or uncheck Tabulate by Precinct in the configuration.

```
SEVERE, "Error reading file!\n%s"
```

Suggested resolution: This error only applies to Clear Ballot files. Ensure that all CVR Files in CVR Files tab are labeled as the correct vendor.

```
SEVERE, "If a cell contains multiple candidates split by the overvote delimiter, it's not valid for any of them to be blank or an explicit undervote."
```

Suggested resolution: Check your CVR file against the data in your election management system and ensure that CVR files are exporting with accurate information.

```
SEVERE, "Error parsing cast vote record:\n%s"
```

Suggested resolution: Confirm that all CVR Files are set to the correct vendor. Check for other error messages in the operator log to see more detailed information about errors. Compare CVR file to expected CVR files available from your vendor to confirm that files are in the correct format.

```
SEVERE, "overvoteDelimiter and overvoteLabel can't both be supplied."
```

Suggested resolution: Review configuration file settings. Ensure that only one of these two fields is filled out on the CVR Files tab.

```
SEVERE, "overvoteDelimiter is required for an ES&S CVR source when overvoteRule is set to "Exhaust if Multiple Continuing"
```

Suggested resolution: Review configuration file settings. Ensure that the overvote delimiter field on the CVR files tab is filled in with the character used in your CVR File to differentiate between candidates ranked at an overvote.

```
SEVERE, "If a cell contains multiple candidates split by the overvote delimiter, it's not valid for any of them to be blank or an explicit undervote."
```

Suggested resolution: Review your CVR. Make sure that it correctly includes data for any overvoted ranking. Check with the CVR vendor to make sure the CVR is exporting correctly from their system.

```
SEVERE, "overvoteDelimiter is invalid."
```

Suggested resolution: Check the information input into the overvote delimiter setting on the CVR Files tab. That setting can't contain any backslashes (`\`) and it must contain at least one character that's not a letter, number, dot, apostrophe, comma, hyphen, quotation mark, or space. For example, an overvote delimiter could be the character `|`. It could never be the character `\`.

```
SEVERE, "Cast vote record identifier missing on row %d in file %s. This may be due to an incorrectly formatted xlsx file. Try copying your cvr data into a new xlsx file to fix
this.",cvrIndex + firstVoteRowIndex, excelFileName
```

Suggested resolution: In addition to the resolution suggested in the message, check your cast vote record and compare it to any expected CVR data from your voting system vendor. Check with the CVR vendor to make sure the CVR is exporting correctly from their system. Users can also remove any CVR ID labels on the CVR Files tab, which should resolve this issue.

```
SEVERE, "No header row found in cast vote record file: %s", this.cvrPath
SEVERE, "No choice columns found in cast vote record file: %s", this.cvrPath
SEVERE: "Wrong number of choice header fields in cast vote record file: %s", this.cvrPath
SEVERE: "Current snapshot has no CVRContests."
SEVERE, "Error parsing contest manifest:\n%s", exception
SEVERE, "Error parsing precinct manifest:\n%s", exception
SEVERE, "Error parsing candidate manifest:\n%s", exception
SEVERE, "No precinct data found!"
SEVERE, "No precinct portion data found!"
SEVERE, "No contest data found!"
SEVERE, "No candidate data found!"
SEVERE, "Precinct ID \"%d\" from CVR not found in manifest data!",precinctId
SEVERE, "Precinct portion ID \"%d\" from CVR not found in manifest data!",precinctPortionId
SEVERE: "GpUnit \"%s\" for CVR \"%s\" not found!", cvr.BallotStyleUnitId, cvr.UniqueId
SEVERE: "GpUnit \"%s\" not found!", unitId
SEVERE: "No Rank found on CVR \"%s\" Contest \"%s\"!", cvr.UniqueId,contest.ContestId
```

Suggested resolution: These are all errors that appear when a CVR file is incompatible with RCTab. Check your cast vote record and compare it to any expected CVR data from your voting system vendor. Check with the CVR vendor to make sure the CVR is exporting correctly from their system. Make sure your voting system version is compatible with RCTab by reviewing the CVR documentation provided with RCTab and by checking with your voting system vendor. If you are unable to resolve the issue this way, file an issue on the RCTab GitHub and we will investigate the problem.

```
SEVERE, "No cast vote record data found!"
```

Suggested resolution: Check your cast vote record and compare it to any expected CVR data from your voting system vendor. Check with the CVR vendor to make sure the CVR is exporting correctly from their system.

```
SEVERE, "Unknown contest ID '%d' found while parsing CVR!", contestId
SEVERE: "Contest \"%s\" from config file not found!", this.contestId
```

Suggested resolution: Double-check the Contest ID for the contest you’re trying to tabulate. Make sure the correct contest ID is included in the config.

```
SEVERE, "Aborting conversion due to cast vote record errors!"
```

Suggested resolution: Check the Operator Log for other messages. Those messages will indicate specific errors with the cast vote record file.

```
SEVERE: "ContestSelection \"%s\" from CVR not found!", contestSelectionId
SEVERE: "CandidateSelection \"%s\" has no CandidateIds!", contestSelection.ObjectId
SEVERE: "CandidateId \"%s\" from ContestSelectionId \"%s\" not found!",contestSelection.CandidateIds[0], contestSelection.ObjectId
```

Suggested resolution: These errors appear if a CDF CVR indicates that a vote ranked a candidate (a contest selection) that the CDF file does not otherwise identify. Check with your vendor to make sure their CDF CVR export matches up with the CDF CVR requirements.

```
SEVERE, "Unexpected error parsing source file: %s\n%s", cvrPath, exception
```

Suggested resolution: RCTab is unable to read the CVR file used. Make sure your voting system version is compatible with RCTab by reviewing the CVR documentation provided with RCTab and by checking with your voting system vendor. If you are unable to resolve the issue this way, file an issue on the RCTab GitHub and we will investigate the problem.

Candidate Errors

Errors in this section are correctable on the Candidates tab. They may also require detailed examination of CVR Files themselves.

```
SEVERE, "Unrecognized candidate found in CVR: %s", candidateId
SEVERE, "Source file contains unrecognized candidate(s): %s"
SEVERE, "Error processing candidate data:\n%s"
SEVERE, "Unrecognized candidate \"%s\" appears %d time(s)!"
SEVERE, "A name is required for each candidate!"
```

Suggested resolution: Add unrecognized candidate names to the Candidates list in the configuration. Error log will list candidate names that do not appear in the CVR files.

```
SEVERE, "Duplicate candidate %ss are not allowed: %s"
```

Suggested resolution: Delete all but one listing of each candidate name in the Candidates list. Candidates only need to be listed once in the Candidate list. Ensure that candidates with similar names are labeled distinctly in the CVR data.

```
SEVERE, "Contest config must contain at least 1 declared candidate!"
```

Suggested resolution: Confirm that candidate names have been added to the candidate list in the Candidates tab.

```
SEVERE, "Error parsing candidate data: %s"
```

Suggested resolution: Re-enter candidate data on the Candidates tab. Confirm that correct CVR files are directed to on CVR files tab.

```
SEVERE, "Contest config must contain at least 1 non-excluded candidate!"
```

Suggested resolution: Confirm that at least one candidate is not excluded on the Candidates tab. If all candidates are excluded, remove excluded candidates. Re-enter candidate names and do not exclude every candidate.

```
SEVERE, "If candidate codes are used, a unique code is required for each candidate!"
```

Suggested resolution: Double-check on the candidates tab that no candidate code values repeat. If code values do repeat, delete that candidate information and re-enter candidate information with a different candidate code.

```
SEVERE, "Candidate code '%s' is not valid for contest '%d'!", candidateCode,contestId
```

Suggested resolution: Check the candidate codes on the "Candidates" tab and cross-reference them with the candidate codes in the contest you’re trying to tabulate. Make sure that all candidate codes are accurate.

Winning Rules Errors

Errors in this section can be corrected on the Winning Rules tab.

```
SEVERE, "If numberOfWinners is zero, winnerElectionMode must be multiSeatBottomsUpPercentageThreshold"!
SEVERE, "numberOfWinners must be zero if winnerElectionMode is multiSeatBottomsUpPercentageThreshold "!"
SEVERE, "If numberOfWinners is zero, multiSeatBottomsUpPercentageThreshold must be specified!"
SEVERE, "nonIntegerWinningThreshold can't be true when winnerElectionMode is \"%s\"!",winnerMode
```

Suggested resolution: Ensure that the correct winner election mode is selected, according to your local RCV laws. After confirming the correct winner election mode, rebuild the configuration file from scratch using the RCTab UI. The settings indicated by these errors cannot be changed or set to incorrect values if a configuration file is created using the UI.

```
SEVERE, "Tabulation can't proceed because all declared candidates are below the minimum vote threshold."
```

Suggested resolution: There are multiple potential solutions here. Check the minimum vote threshold setting for the contest - is it set according to your local tabulation rules? If it is, then no candidate has sufficient votes to win in this contest. Refer to your local law for what to do next. If the minimum vote threshold setting is out of sync with your local tabulation rules, reset it to the correct setting and run tabulation again.

```
SEVERE, "When tiebreakMode involves a random element, randomSeed must be supplied."
```

Suggested resolution: Input a positive or negative number into the random seed box on the Winning Rules tab.

Voter Error Tab Errors

Errors in this section can be corrected on the Voter Error tab.

```
SEVERE, "When overvoteLabel is supplied, overvoteRule must be either "exhaustImmediately " + "or alwaysSkipToNextRank!"
SEVERE, When overvoteLabel is supplied, overvoteRule must be either "Always skip to next rank" or "Exhaust immediately"!
```

Suggested resolution: Check what overvote rule is selected on the Voter Error tab. Ensure that it is the correct rule based on your local RCV laws. If either of the two rules mentioned in this error are required by your local laws, select the correct one.

Results Writing Errors

Errors in this section relate to RCTab's ability to save results data after a tabulation. They can be corrected by ensuring that RCTab has the ability to read/write to disk, that there is sufficient disk space to create files, and by checking that valid filepaths are indicated in the configuration file.

```
SEVERE, "Error writing summary files:\n%s"
```

Suggested resolution: Ensure that your computer has sufficient space to store summary files and check that RCTab has write permission for the location selected for file storage on the Output tab.

```
SEVERE, "Error saving file: %s\n%s"
```

Suggested resolution: Confirm that the location you have set files to save is a correct filepath.

```
SEVERE, "Failed to create output directory: %s\n" + "Check the directory name and permissions"
```

Suggested resolution: Confirm that the location you have set files to save to is a correct filepath and that RCTab has permissions in that location.

```
SEVERE, "Error writing to JSON file: %s\n%s\nPlease check the file path and permissions!"
SEVERE, "Error creating CSV file: %s\n%s\nPlease check the file path and permissions!"
SEVERE, "Error saving file: %s\n%s"
```

Suggested resolution: Check file path on output file settings. Confirm file path is correct. Also confirm that RCTab has permission to write to the filepath selected.

```
SEVERE, "CDF JSON generation failed."
```

Suggested resolution: Uncheck CDF JSON in your configuration.

```
SEVERE, "Error writing cast vote records in generic format from input file: %s\n%s"
```

Suggested resolution: This error only applies to certain CVR files. Ensure that all CVR files in the CVR Files tab are labeled as the correct vendor and that RCTab has read/write permissions for the location on the drive where you are attempting to save files.

System Errors

Errors in this section relate to RCTab's ability to actually run tabulations and potential errors in booting the software.

```
SEVERE, "Failed to configure tabulation logger!\n%s"
SEVERE, "Failed to configure logger!"
```

Suggested resolution: Close and reboot RCTab.

```
SEVERE, "Error opening file: %s\n%s\n" + "Check file path and permissions and make sure they are correct!"
```

Suggested resolution: Confirm that the location you have set files to save is a correct filepath.

```
SEVERE, "RCTab security error. Could not meet FIPS compliance."
SEVERE, "Error checking if output directory was in Users directory"
```

Suggested resolution: Restart RCTab and confirm that you are running with local administrator permissions.

```
SEVERE, "Failed to open: %s:\n%s."
```

Suggested resolution: Close RCTab and re-launch according to the relevant launch instructions in **Section 18 - User Guide** document.

```
SEVERE, "User exited tabulator before it was finished!"
SEVERE, "Tabulation was cancelled by the user!"
```

Suggested resolution: Re-launch tabulator. Run Tabulation again and do not close RCTab.

```
SEVERE, "Error loading text file: %s\n%s",configFileDocumentationFilename,exception
```

Suggested resolution: Close RCTab and re-launch according to the relevant launch instructions in **Section 18 - User Guide** document.

```
SEVERE, "Error during tabulation:\n%s\nTabulation failed!"
```

Suggested resolution: Check operator error log for other error messages. Other error messages will appear explaining why Tabulation could not be completed.

```
SEVERE, "Failed to open: %s:\n%s. "
```

Suggested resolution: Close RCTab and re-launch according to the relevant launch instructions in **Section 18 - User Guide** document.

```
SEVERE, "Files are unequal lengths!"
SEVERE, "Files are not equal (line %d):\n%s\n%s"
SEVERE, "File not found!\n%s"
SEVERE, "Error reading file!\n%s"
SEVERE, "Error closing file!\n%s"
SEVERE, "Error deleting file: %s\n%s"
```

Suggested resolution: These are errors presented by code when an automated test fails. If a user is seeing these errors, restart RCTab and build a new configuration file for the contest you are trying to tabulate.

Other errors

Errors in this section do not fit neatly into any other sections in this document.

```
SEVERE: "No config file path provided on command line!Please provide a path to the config file!See README.md for more details."
```

Suggested resolution: Use the user interface to interact with RCTab, or check that you have followed **Section 24 - Command Line Instructions** properly.

2.34.2 WARNING MESSAGES

```
WARNING: "winnerElectionMode \"%s\" is unrecognized! Please supply a valid winnerElectionMode.", oldWinnerElectionMode

WARNING: "tiebreakMode \"%s\" is unrecognized! Please supply a valid tiebreakMode.",oldTiebreakMode

WARNING: "CVR has no adjudicated rankings, skipping: Tabulator ID: %s Batch ID: %s Record ID: %d",tabulatorId, batchId, recordId

WARNING: "Precinct identifier not found for cast vote record: %s", computedCastVoteRecordId

WARNING: "Unexpected XML data: %s %b %s", s, b, s1

WARNING: "Invalid selection! Please try again."

WARNING: "Please load a contest config file before attempting to tabulate!"

WARNING: "Please load a contest config file before attempting to convert to CDF!"

WARNING: "Unable tell if saving is necessary, but everything should work fine anyway! Prompting for save just in case...\n%s",exception

WARNING: "Unable to set emptyConfigString, but everything should work fine anyway!\n%s",exception

WARNING: "CandidateSelection \"%s\" has multiple CandidateIds. Only the first one will be processed.",contestSelection.ObjectId

WARNING: "CandidateSelection \"%s\" has multiple CandidateIds. Only the first one will be processed.",contestSelectionId
```

2.34.3 INFO MESSAGES

```
INFO: "Tabulator is being used via the CLI."

INFO: "Launching %s version %s...", APP_NAME, APP_VERSION

INFO: "Round: %d", currentRound

INFO: "Candidate \"%s\" was elected with a surplus fraction of %s.",winner, surplusFraction

INFO: "Randomly generated candidate permutation for tie-breaking:"

INFO: "%s", candidateId

INFO: "%s had residual surplus of %s.", winner, winnerResidual

INFO: "Winning threshold set to %s.", winningThreshold

INFO: "Parsed %d records from %d files", recordsParsed, cvrSequence

INFO: "Parsed %d cast vote records.", recordsParsed

INFO: "Successfully loaded contest config: %s", configPath

INFO: "Validating contest config..."

INFO: "Contest config validation successful."

INFO: "Creating new contest config..."

INFO: "Exiting tabulator GUI..."

INFO: "Opening tabulator GUI..."

INFO: "Welcome to %s version %s!", Main.APP_NAME, Main.APP_VERSION

INFO: "Contest Selection needs adjudication. Skipping."

INFO: "Starting tabulation session..."

INFO: "Config file: %s", configPath

INFO: "This is a multi-pass IRV contest."

INFO: "Excluding %s from the remaining tabulations.", newWinner

INFO: "Tabulation session completed."

INFO: "Results written to: %s", outputPath
```

```
INFO: "Parsing cast vote records..."
INFO: "Reading CDF cast vote record file: %s...", cvrPath

INFO: "Reading Clear Ballot cast vote records from file: %s...", cvrPath

INFO: "Reading Dominion cast vote records from folder: %s...", cvrPath

INFO: "Reading ES&S cast vote record file: %s...", cvrPath

INFO: "Reading Hart cast vote records from folder: %s...", cvrPath

INFO: "Check config settings for candidate names, firstVoteRowIndex, firstVoteColumnIndex, and precinctColumnIndex to make sure they are correct!"

INFO: "See config_file_documentation.txt for more details."

INFO: "Check file path and permissions and make sure they are correct!"

INFO: "ES&S cast vote record files must be Microsoft Excel Workbook format.\nStrict Open XML and Open Office are not supported."

INFO: "See the log for details."

INFO: "JSON file generated successfully."

INFO: "Generating summary spreadsheet: %s...", csvPath

INFO: "Summary spreadsheet generated successfully."

INFO: "Writing cast vote records in generic format to file: %s...",
outputPath

INFO: "Generating cast vote record CDF JSON file: %s...", outputPath

INFO: "Generating summary JSON file: %s...", jsonPath

INFO: "Migrated tabulator config version from %s to %s.",config.rawConfig.tabulatorVersion != null ? config.rawConfig.tabulatorVersion : "unknown",Main.APP_VERSION
```

2.35 Section 30 - RCTab System Tab Hints

These hints are displayed in the user interface of RCTab to help users navigate through each option available in RCTab.

2.35.1 Hints for Contest Info Tab

The tabulator calculates results for one contest at a time, rather than the results of several contests for an election all at once. The information on this tab is for the particular contest you will be tabulating.

These fields do not influence the computations. They are shown in the final output file(s) to help connect the data (results) with the contest the results belong to.

- Think long term, e.g. from the perspective of looking at the results files 6 months after the election and wanting to be clear what contest the results belong to.
- You may find it helpful to revisit this tab once you have done a few test runs and see what the output looks like.

Contest Name (required): Enter a name to identify it.
Examples: City Council 2018, Board of Election Ward 13 2017, Mayor, Referendum 289b

Contest Date (optional): The date on which the election for this contest was run.

Contest Jurisdiction (optional): E.g.: Minneapolis, Eastpointe
Whether this is helpful may depend on what you put into the Contest Name field

Contest Office (optional): E.g.: Mayor, County Clerk
Whether this is helpful may depend on what you put into the Contest Name field

Rules Description (optional): What short description of this configuration would help you remember in, say, six months what election this specific rule configuration is for?

2.35.2 Hints for Candidates Tab

Fill in the fields and click the Add button:

Name (required): E.g.: Dave Harris.

Code (optional): Some CVR files use codes in lieu of full candidate name. E.g.: "JCD" or "14".

Excluded (optional): When checked, the candidate will be ignored during tabulation. An example of when this might be used: a candidate dropped out after the ballots were printed.

2.35.3 Hints for CVR Files Tab

The tabulator needs to know where each of your CVR files is and how to interpret each of them. As you add files, it will build up a list of files to use when it tabulates the results of this contest.

For each of your CVR files, provide the necessary information and then use the Add button to add it to the list.

Provider (required): The vendor/machine that generated (produced) the file. After you select the field, the tabulator will fill in as many of the other fields as it can based on what it knows about that provider. You can adjust those values as necessary.

Path (required): Location of the CVR file.

- Example: /Users/test data/2015-portland-mayor-cvr.xlsx

Contest ID (required for non-ES&S): Some CVRs assign an ID label to each contest in the CVR. The tabulator needs to know which contest is being tabulated when multiple contests are included in one CVR. Enter the ID of the contest being tabulated in this field.

First Vote Column (required for ES&S): the column where the first vote record is.

First Vote Row (required for ES&S): the row where the first vote record is.

ID Column (optional): The column the IDs are in. Not all CVR files contain an ID column.

Precinct Column (required for ES&S if you want to tabulate by precinct): The column that contains the precinct.

Overvote Delimiter (optional, but must be blank if "Overvote Label" is provided): If using a CVR in ES&S style, overvotes can be reflected in a CVR by displaying all candidates marked at a ranking. Those candidate names will be differentiated from each other by a delimiter, something like a vertical bar | or a slash /. If your overvotes are delimited like this, enter the delimiter used in this field. Note that ES&S files may include only the label "overvote" and no additional information, in which case the "Overvote Label" field should be used instead.

Overvote Label (optional): Some CDF and ES&S CVRs use a particular word/phrase to indicate an overvote.

Undervote Label (optional): Some ES&S CVRs use a particular word/phrase to indicate an undervote. Undeclared Write-in Label (optional): Some CVRs use a particular word/phrase to indicate a write-in.

Treat Blank as Undeclared Write-in (optional): When checked, the tabulator will interpret blank cells in this ES&S CVR as votes for undeclared write-ins.

2.35.4 Hints for Winning Rules Tab

Winner Election Mode (required): What process to use for selecting winner(s) for this contest.

- Single-winner majority determines winner: Elects one winner. Eliminate candidates one-by-one or using batch elimination until a candidate emerges with a majority. Candidate with the most votes at the end wins.
- Multi-winner allows only one winner per round: Elects multiple winners. Elect and transfer the surplus vote of only the candidate with the most votes if multiple candidates exceed the winning threshold in a round of counting.
- Multi-winner allows multiple winners per round: Elects multiple winners. Elect and transfer the surplus vote of all candidates crossing the winning threshold if multiple candidates exceed the winning threshold in a round of counting.
- Bottoms-up: Eliminate candidates until the desired number of winners is reached, then stop. Bottoms up does not transfer surplus votes.
- Bottoms-up using percentage threshold: Elects multiple winners. Eliminate candidates until the remaining candidates have a vote share equal to or greater than a specified percentage of the vote.
- Multi-pass IRV: Elects multiple winners. Eliminate candidates one-by-one or using batch elimination until only two candidates remain. Candidate with the most votes at the end wins. Run a new set of rounds with any winning candidates ignored.

Maximum Number of Candidates That Can Be Ranked: How many rankings each voter has in this contest.

Minimum Vote Threshold: The number of first-choice votes a candidate must receive in order to remain in the race. Most jurisdictions do not set a minimum vote threshold.

Use Batch Elimination: Batch elimination, or simultaneous elimination of all candidates for whom it is mathematically impossible to be elected, eliminates all candidates who cannot receive enough votes to surpass the candidate with the next highest number of votes. Example: in a six candidate contest with 200 votes, Candidate A has 80 votes, Candidate B has 70, and the other four combined have 50. Because those four candidates can never combine their votes to surpass Candidate B, they can be batch eliminated. Available only when Winner Election Mode is "Single-winner majority determines winner".

Continue until Two Candidates Remain: Single-winner ranked-choice voting elections can stop as soon as a candidate receives a majority of votes, even though 3 or more candidates may still be in the race. Selecting this option will run the round-by-round count until only two candidates remain, regardless of when a candidate wins a majority of votes.

Tiebreak Mode (required): Ties in ranked-choice voting contests can occur when eliminating candidates or when electing candidates. Multi-winner contests can have ties between candidates who have both crossed the threshold of election; in that case ties are broken to determine whose surplus vote valuetransfers first. Tiebreak procedures are set in law, either in the ranked-choice voting law used in your jurisdiction or in the elections code more generally. Select the option from this list that complies with law and procedure in your jurisdiction.

- Random: Randomly select a tied candidate to eliminate or, in multi-winner contests only, elect. Requires a random seed.
- Stop counting and ask: Pause count when a tie is reached. User is prompted to select any tied candidate to eliminate or, in multi-winner contests only, elect.
- Previous round counts (then random): The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, the tie is broken randomly. Requires a random seed.
- Previous round counts (then stop counting and ask): The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, user is prompted to select any tied candidate to eliminate or, in multi-winner contests only, elect.
- Use candidate order in the config file: Use the order of candidates in the config file to determine tiebreak results. Candidates lower in the list lose the tiebreaker.
- Generate permutation: Generate a randomly ordered list of candidates in the contest. Candidates lower in the permutation lose the tiebreaker. Requires a random seed.

Random Seed (required if Tiebreak Mode is "Random", "Previous round counts (then random)", or "Generate permutation"): Enter a positive or negative integer to generate random orders.

Number of Winners: The number of seats to be filled in the contest.

Percentage Threshold: The share of votes a candidate must have in order to win. Candidates falling below this threshold are eliminated one-by-one beginning with the candidate with the fewest votes. Available only when Winner Election Mode is "Bottoms-up using percentage threshold".

Threshold Calculation Method: The threshold of election is the number of votes a candidate must receive in order to win election. There are three primary ways to calculate the threshold of election in multi-winner RCV contests. This will be set in law (either by statute or regulation) in your jurisdiction. Available only when Winner Election Mode is "Multi-winner allow only one winner per round" or "Multi-winner allow multiple winners per round".

- Compute using most common threshold formula: The most common threshold formula is calculated by dividing the number of votes by the number of seats plus one, then adding one to that number. Fractions are disregarded. This is also known as the Droop quota. Candidates must receive this number of votes (or more) to win.
- Compute using HB Quota: The HB, or Hagenbach-Bischoff, Quota divides the number of votes by the number of seats plus one, leaving fractions. Candidates must receive more than this number of votes to win.
- Compute using Hare Quota: The Hare quota divides the number of votes by the number of seats. It requires candidates to receive that number of votes (or more) to win.

Decimal Places for Vote Arithmetic (Multi-Winner Only): Sets how many decimal places after the decimal point are used in surplus transfers and in calculating the threshold.

2.35.5 Hints for Voter Error Rules Tab

The tabulator needs to know how to handle voter errors in your jurisdiction. These requirements are typically included in statute or regulation.

Overvote Rule (required): How to handle a ballot where a voter has marked multiple candidates at the same ranking when that ballot is encountered in the round-by-round count.

- Always skip to next rank: Skips over an overvote and goes to the next validly-marked ranking on a ballot.
- Exhaust immediately: A ballot with an overvote exhausts when that overvote is encountered in the rounds of counting.
- Exhaust if multiple continuing: If a voter has an overvote but only one candidate at that overvote is still in the race when that overvote is encountered, the ballot counts for that candidate. If multiple candidates at the overvote are still in the race, the ballot exhausts.

How Many Consecutive Skipped Ranks Are Allowed (required): How many rankings in a row can a voter skip and still have later rankings count? 0 allows no skipped rankings. 1 allows voters to skip rankings one at a time, but not more than 1 in a row, and so on.

Example: A voter could rank in 1, 3, 5 and not exhaust under this rule, for example.

Exhaust on Multiple Ranks for the Same Candidate: When checked, the tabulator will exhaust a ballot that includes multiple rankings for the same candidate when that repeat ranking is reached.

Example: A voter ranks the same candidate 1st and 3rd, a different candidate 2nd, and another candidate 4th. If their original first choice and their second choice are eliminated, the ballot exhausts when it reaches the repeat ranking in rank 3. The ranking in the 4th rank does not count.

2.35.6 Hints for Output Tab

Tell the tabulator where results files go and what additional results files you want.

Output directory: The location where results files will go. If no value (or a relative path, like "output") is supplied, the location where the config file is saved will be used as the base directory. Absolute paths, like "C:\output" work too.

Tabulate by Precinct: Produce round-by-round results at the precinct level. Results are how ballots at the precinct level transferred in the contest as a whole, not a simulated round-by-round count in the precinct. Requires precinct information in CVR Files tab.

Generate a CDF JSON: Produce a VVSG common data format JSON file of the CVR.

2.36 Section 31 - Coding and Header Comment Standards for RCTab

The BrightSpots programmers that do the coding of the RCTab following the Google Java Style Guide. They do not have revision histories on individual files because comprehensive file revision histories between versions are easily viewable on GitHub.

The CVVS includes the following Header Comments requirements:

5.2.6 Header Comments

Header comments and other commenting standards should be specified by the selected coding standard in a manner consistent with the idiom of the programming language chosen. If the coding standard specifies a coding style and commenting standard that make header comments redundant, then they may be omitted. Otherwise, in the event that the coding standard fails to specify the content of header comments, application logic modules should include header comments that provide at least the following information for each callable unit (function, method, operation, subroutine, procedure, etc.):

a. The purpose of the unit and how it works (if not obvious);

b. A description of input parameters, outputs and return values, exceptions thrown, and side-effects;

c. Any protocols that must be observed (e.g., unit calling sequences);*

d. File references by name and method of access (read, write, modify, append, etc.);

e. Global variables used (if applicable);

f. Audit event generation;

g. Date of creation; and

h. Change log (revision record). Change logs need not cover the nascent period, but they must go back as far as the first baseline or release that is submitted for testing, and should go back as far as the first baseline or release that is deemed reasonably coherent.

The coding style and commenting standard used in developing RCTab, the Google Java Style Guide, makes header comments redundant. Per the statement in 5.2.6 that if a "coding standard specifies a coding style and commenting standard that make[s] header comments redundant," we exclude extensive header comments in source code. One of the benefits of writing high-quality code is that there's less need to add extraneous comments since the code speaks for itself.

The Google Java Style Guide is attached as an addendum to this document.

2.37 Section 32 - Secure USB Process

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.

2.37.1 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.

2.37.2 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.

2.37.3 Setting a PIN on datAshur PRO Memory Stick

Once the datAshur PRO Memory Stick has been formatted, the manufacturer will program a user PIN 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. (`3-3-3-3-3-3-3`)
- 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.

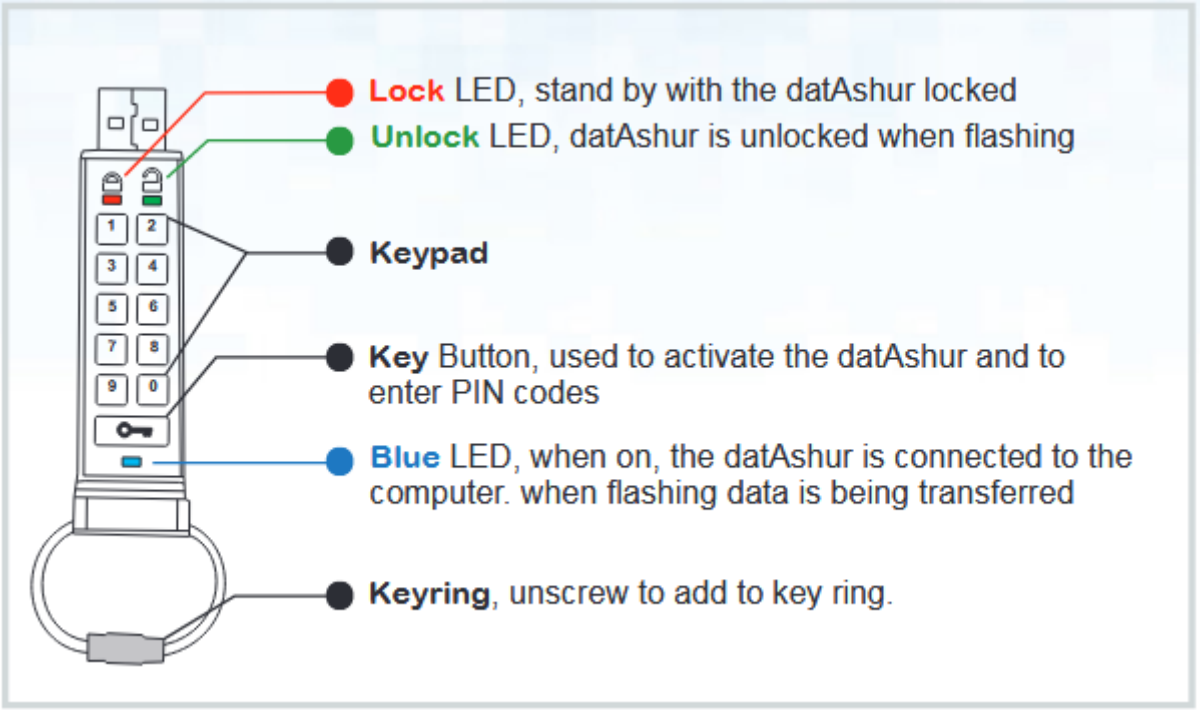
5. Re-enter the new user PIN and press the key button twice. The red LED will illuminate and then change to a solid green LED if PIN entries match.
6. Once the datAshur PRO Memory Stick locks, enter the new PIN and verify it unlocks the memory stick.

a. After the PIN has been changed, the default PIN (`1-1-2-2-3-3-4-4`) will no longer work.

2.37.4 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.

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

2.38 Appendix I - Expected Outcome RCV Test Sets Multi-Winner

Content for Expected Outcome RCV Test Sets Multi-Winner.

2.39 Appendix II - Expected Outcome RCV Test Sets Single-Winner

Content for Expected Outcome RCV Test Sets Single-Winner.

2.40 Appendix III - Ranked Choice Voting Laws

Content for Ranked Choice Voting Laws.

2.41 Appendix IV - 22-Month Archiving Procedure

Content for 22-Month Archiving Procedure.

3. User Guide

3.1 User Guide

Users should ensure instructions below have been followed and completed prior to operating RCTab:

- **Section 05 - Acceptance Test Procedures**
- **Section 16 - System Hardening Procedures - Windows OS**
- **Section 22 - Installation Instructions for Windows OS**
- **Section 23 - HashCode Instructions - Windows OS**
- **Section 25 - Configuration File Parameters**

Any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team with no less than two trained personnel. This document describes all interfaces and options in the RCTab software.

3.1.1 Launching RCTab

The manufacturer recommends RCTab be installed as part of the pre-election preparation process. In order to determine the appropriate launch procedure for a jurisdiction, users should consider the maximum possible number of votes that could occur in the event all eligible voters presented themselves to vote. Jurisdictions should, as part of their pre-election procedure, launch RCTab according to the relevant launch instructions described below to ensure they launch it in accordance with the below requirements. The manufacturer is available for support with this process. For acceptance testing and L&A procedures to set up and use of RCTab, see:

- **Section 05 - Acceptance Test Procedures**
- **Section 11 - L&A Testing**

Contests with fewer than 1,000,000 votes

To Launch:

1. Navigate to the rcv folder created when you unzipped RCTab.
2. Open the bin folder
3. Right-click on the `rcv.bat` file. Click "Run as Administrator". If a "Windows protected your PC" window pops up click "More Info" then click the "Run anyway" button. Enter the administrator password

Contests with more than 1,000,000 votes

1. Open a Command Prompt by navigating to the start menu and typing in Command Prompt.
2. Press enter to launch Command Prompt.
3. Change the current directory to the `rcv` folder created when you unzipped RCTab.
4. First, type in cd (note there should be a space after cd).
5. Using File Explorer navigate to the folder where RCTab is installed.
6. Double-click on the `rctab_v1.3.0_windows` folder
7. Click and drag the `rcv` folder over to the command prompt window
8. Your command prompt will now read something like:

a. `cd C:\RCTab\rctab_v1.3.0_windows\rcv`

9. Press enter

10. Launch the tabulator by entering the following command:

a. `.\bin\java -mx30G -p .\app -m network.brightspots.rcv/network.brightspots.rcv.Main`.

3.1.2 Prepping CVRs for use with RCTab Software

All exports of CVR records for use with RCTab and migration of CVR records for use with RCTab should follow CVR procedures required by the jurisdiction. Users must keep track of the path to each file that is needed to tabulate the RCV contests. This will be needed when configuring the RCTab software.

3.1.3 Setting up a configuration file for RCTab

All settings in the RCTab software are described in technical detail in **Section 25 - Configuration File Parameters**, including brief descriptions of how options impact the operational validity of other options. This document will describe how to set up a configuration for a contest using your jurisdiction's ranked choice voting rules. This guide also includes screenshots of the interfaces described. The values a user inputs into any of these fields depend upon the relevant laws and regulations in place in their jurisdiction, as well as the voting system vendor used to produce cast-vote records for their elections. Users must understand the requirements of their laws, regulations, and vendor CVR data in order to fill out these fields accurately for their needs.

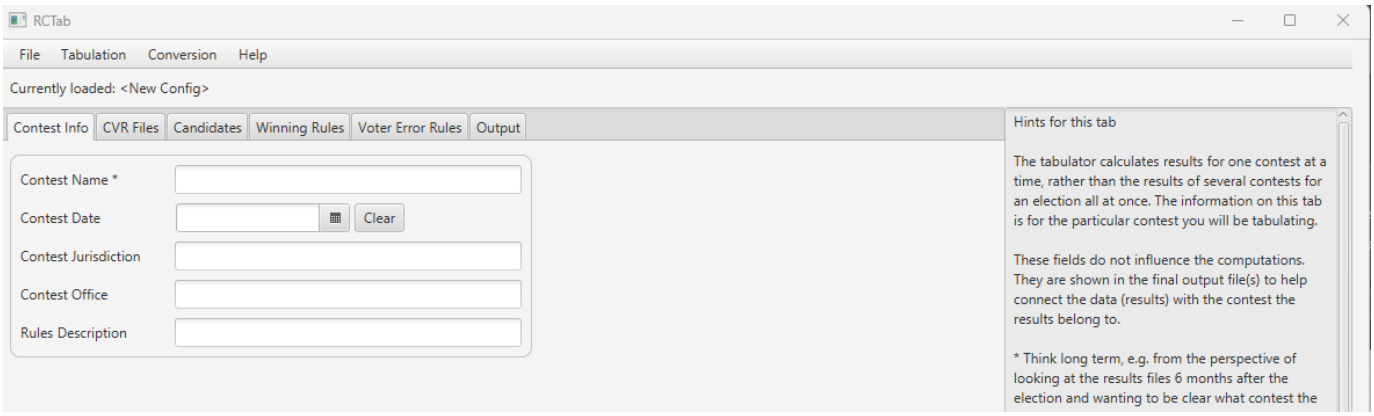
RCTab is organized into tabs: the Contest Info Tab, the CVR Files Tab, the Candidates Tab, the Winning Rules Tab, the Voter Error Rules Tab, and the Output Tab. These tabs each include a set of fields that users fill out - some fields are always required, some fields are required based on previous input, and some fields are always optional. This guide will take the user through basic descriptions of each of these tabs and each of these fields. Additional technical information about these fields can be found in **Section 25 - Configuration File Parameters**. As users navigate through each tab they are building a configuration `.json` file which must be saved when complete. If RCTab is closed and the configuration is not saved, no information will be stored. The next time RCTab is opened all fields will be blank. Once a configuration is saved, it can be used by RCTab to process RCV election results according to the rules laid out in that configuration, provided all rules necessary are filled out.

Each individual contest run through RCTab requires its own configuration with contest-specific information such as the contest name and candidate names. This guide will run through setting up the winning rules and voter error rules requirements that will apply in your election and will discuss procedures for how to properly fill out other fields in the software. Operation of RCTab must be conducted by a team of no less than two trained personnel. Note: Including a % symbol anywhere in a configuration file results in a crash of the software. Do not use % symbols in any portion of a configuration file, including settings such as file paths and candidate names.

The following guide describes how to operate RCTab. See also **Section 11 - L&A Testing** document for additional detail on system operations.

Contest Info Tab

Below is what the contest info tab looks like when a user first navigates to it. This is also the first screen a user will see upon successfully launching RCTab. Contest info fields are in the middle of the screen. The black box at the bottom of the screen is called the operation log box. It sends the user messages about the tabulation process, any errors encountered in using a configuration file, any errors in tabulation, and other software errors. Information in this panel is saved to the `rcv_0.log file`. Users cannot turn this feature off while using the RCTab software. The panel on the right side of the document is the “Hints'' tab which includes any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team with no less than two trained personnel.



These settings appear on the "Contest Info" tab in RCTab.

- Contest Name
- Contest Date
- Contest Jurisdiction
- Contest Office
- Rules Description

The tabulator calculates results for one contest at a time, rather than the results of several contests for an election all at once. The information on this tab is for the particular contest you will be tabulating.

These fields do not influence the computations. Contest Name, Contest Date, Contest Jurisdiction, and Contest Office are shown in the final output file(s) to help connect the data (results) with the contest the results belong to.

- Think long term, e.g. from the perspective of looking at the results files 6 months after the election and wanting to be clear what contest the results belong to.
- You may find it helpful to revisit this tab once you have done a few test runs and see what the output looks like.

Contest Name (required): Enter a name to identify it.

- Examples: City Council 2018, Board of Election Ward 13 2017, Mayor, Referendum 289b

Contest Date (optional): The date on which the election for this contest was run.

- Clear permits user to clear this field
- The calendar button allows a user to navigate through a calendar to select the date of the contest

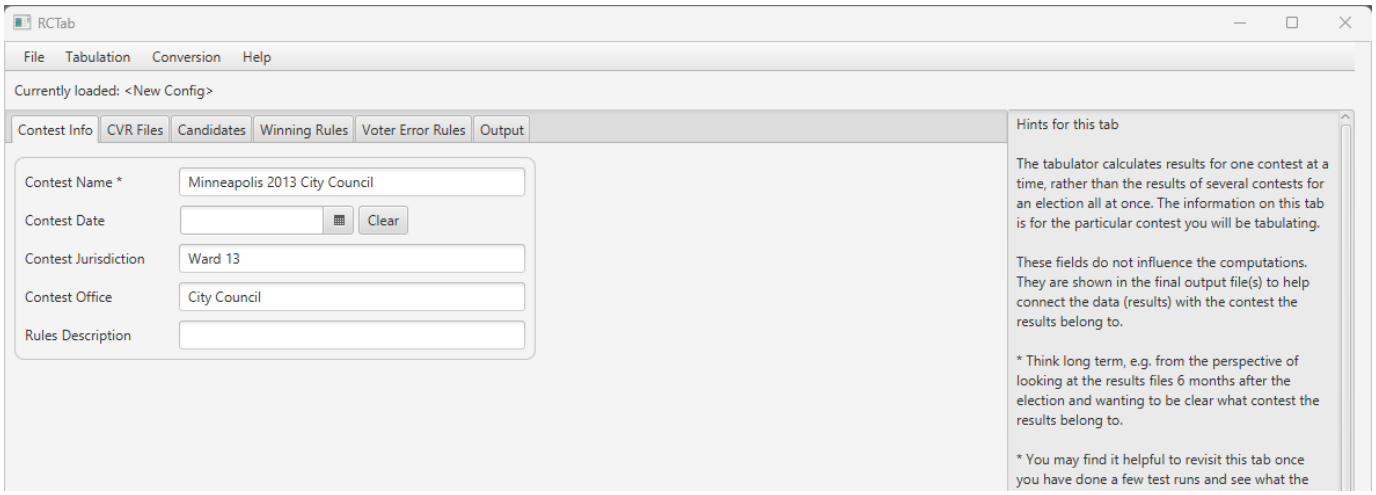
Contest Jurisdiction (optional): E.g.: Minneapolis, Eastpointe

- Whether this is helpful may depend on what you put into the Contest Name field

Contest Office (optional): E.g.: Mayor, County Clerk

- Whether this is helpful may depend on what you put into the Contest Name field

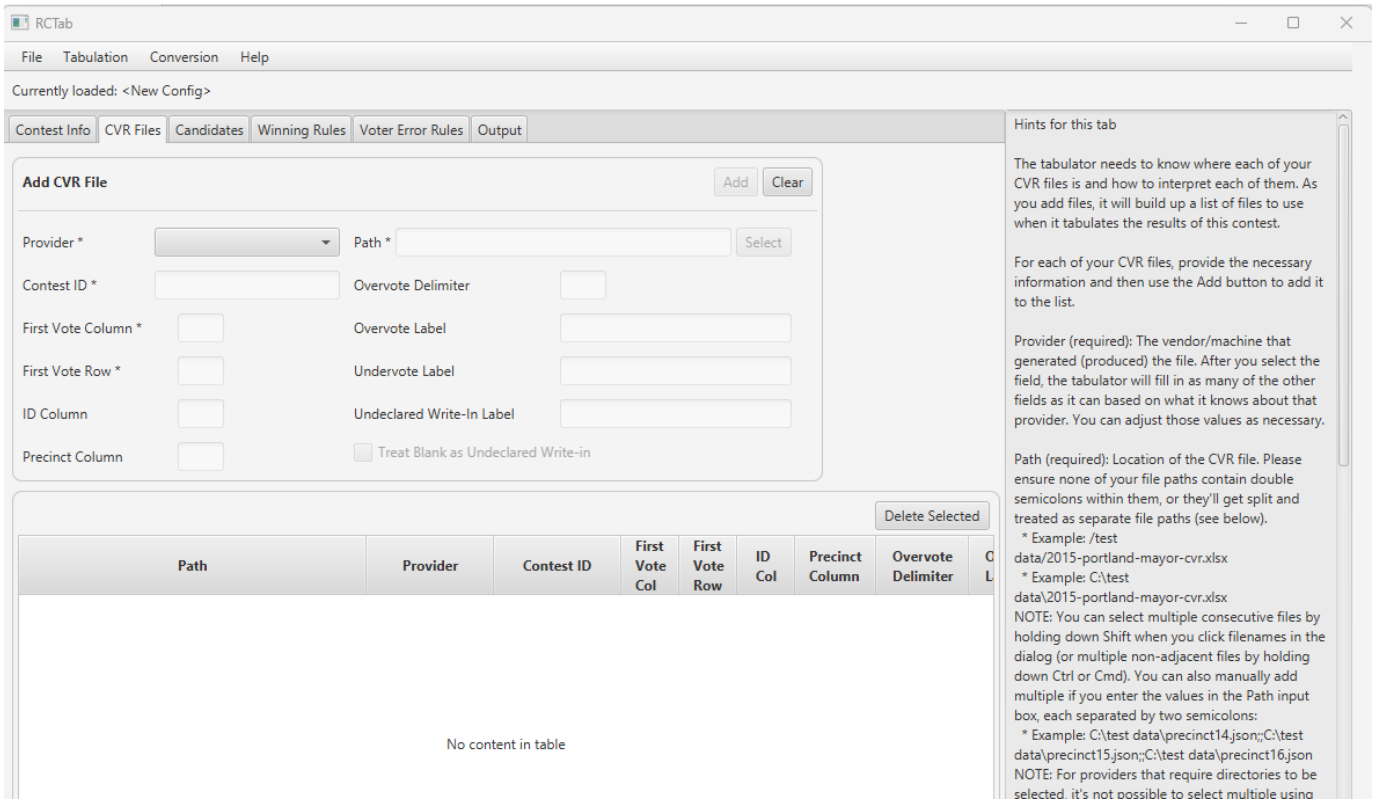
Rules Description (optional): What short description of this configuration would help you remember in, say, six months what election this specific rule configuration is for? This option’s use impacts no other options. It is included in configuration `.json` files and audit `.log` files.



Example Completed Contest Info tab

CVR Files Options

This is what the CVR Files tab looks like when a user first navigates to it:



The tabulator needs a configuration file laying out where each of its user's CVR files for the contest to be tabulated are and how to interpret each of them. Only add files to the configuration that contain data for the contest you are tabulating. CVRs that do not include data for the contest to be tabulated will cause tabulation to fail. Fields used to describe CVRs are on the top half of the tab, while the table on the bottom half of the tab will display each CVR file you have added to your configuration file so far. All information in this tab directs RCTab on where to find files and how to read those files, it does not actually pull in any vote information. Vote information is only used after a user clicks the "Tabulate" button under "Tabulation" as described later in this guide.

For each of your CVR files, provide the necessary information and then use the Add button to add it to the list.

Provider (required): The vendor/machine that generated (produced) the file. After you select the field, the tabulator will fill in as many of the other fields as it can, based on what it knows about that provider. You can adjust those values as necessary. Different options are active for different providers. See below for a break-down of each field and what is required:

Path (required)**:** Location of the CVR file.

- Example: `/Users/test data/2015-portland-mayor-cvr.xlsx`
- Select allows a user to navigate using Windows Explorer to select a location.

Contest ID: Some CVRs assign an ID label to each contest in the CVR. The tabulator needs to know which contest is being tabulated when multiple contests are included in one CVR. Enter the ID of the contest being tabulated in this field.

First Vote Column: the column where the first vote record is.

First Vote Row: the row where the first vote record is.

ID Column: The column the IDs are in. Not all CVR files contain an ID column.

Precinct Column: The column that contains the precinct.

Overvote Delimiter (optional, but must be blank if "Overvote Label" is provided)**:** If using a CVR in ES&S style, overvotes can be reflected in a CVR by displaying all candidates marked at a ranking. Those candidate names will be differentiated from each other by a delimiter, something like a vertical bar `|` or a slash `/`. If your overvotes are delimited like this, enter the delimiter used in this field. Note: that ES&S files may include only the label "overvote" and no additional information, in which case the "Overvote Label" field should be used instead.

Overvote Label (optional, but must be blank if Overvote Rule is)**:** Some CDF and ES&S CVRs use a particular word/phrase to indicate an overvote.

Undervote Label (optional)**:** Some ES&S CVRs use a particular word/phrase to indicate an undervote.

Undeclared Write-in Label (optional)**:** Some CVRs use a particular word/phrase to indicate a write-in.

Treat Blank as Undeclared Write-in (optional for ES&S)**:** When checked, the tabulator will interpret blank cells in this ES&S CVR as votes for undeclared write-ins.

Add: Adds CVR to the configuration file. Impacts no other option. Operation impacted by whether required fields (depending on Provider selected) are filled in.

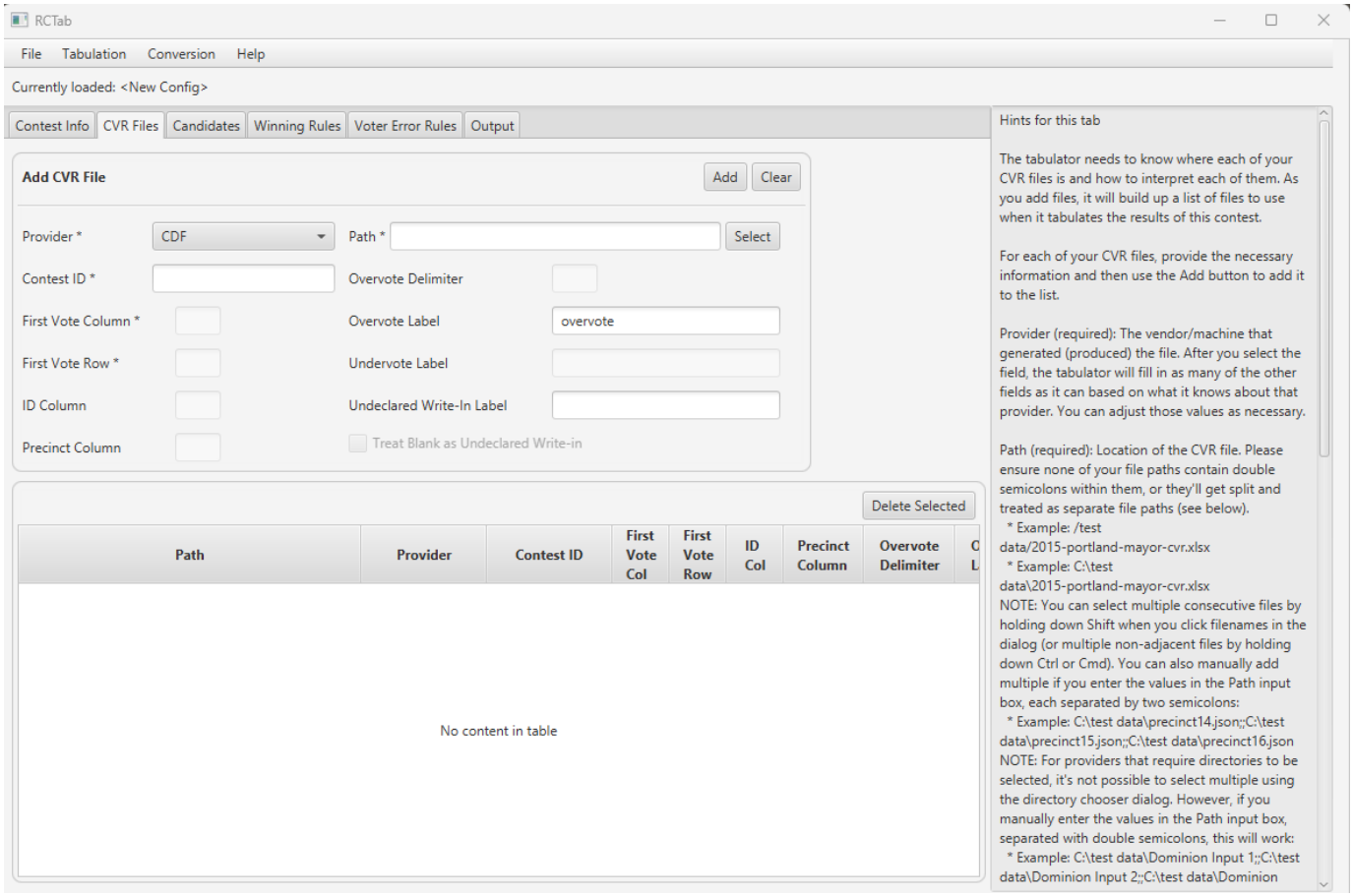
Clear: Clears all fillable values in CVR Files tab above the CVR Files table.

Delete Selected: Deletes CVR file information listed in the CVR table. Impacted by and impacts no options.

This guide will now briefly show which provider settings permit users to edit which additional settings.

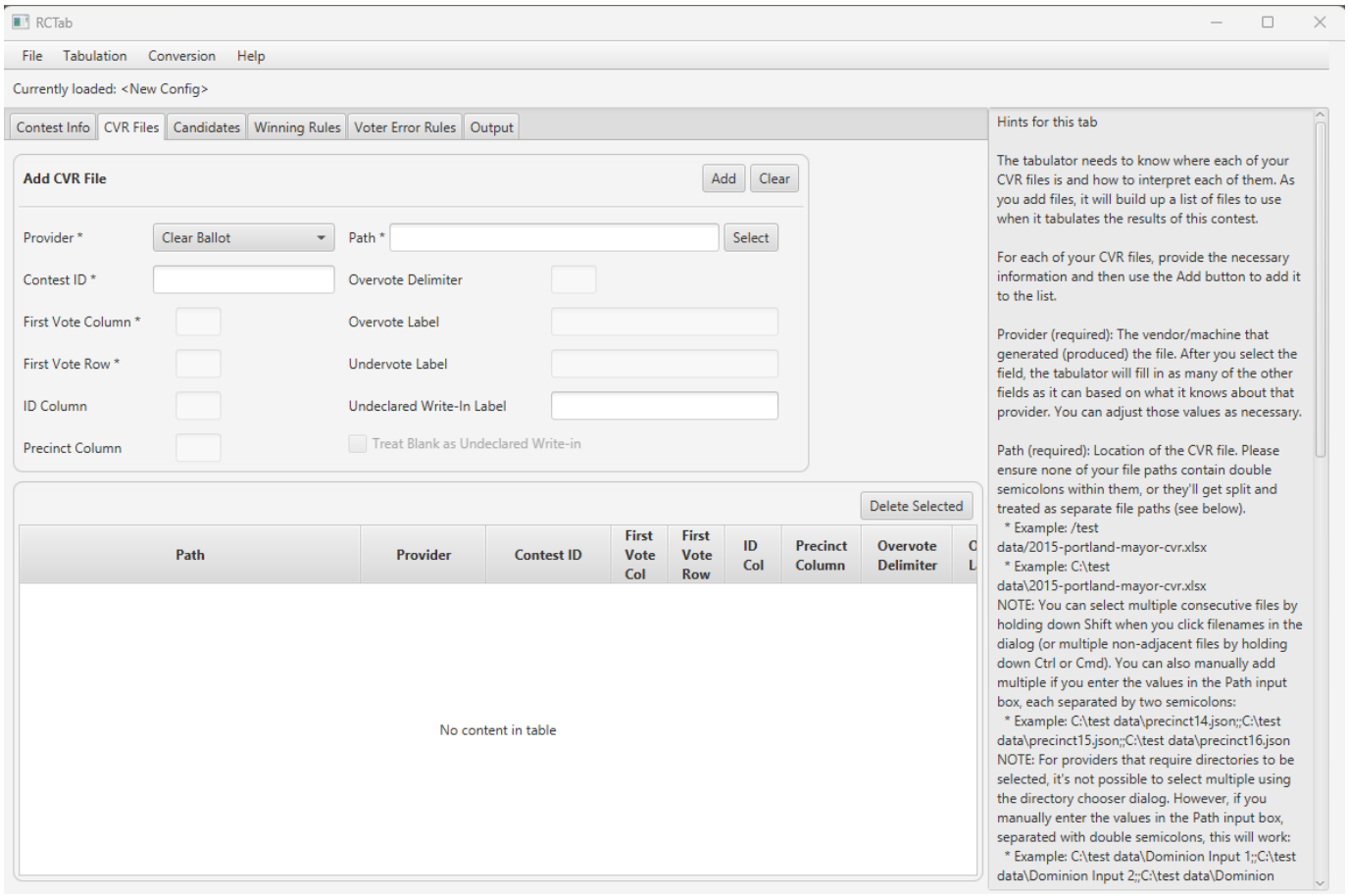
- Provider options:
- CDF
- Clear Ballot
- Dominion
- ES&S
- Hart

CDF



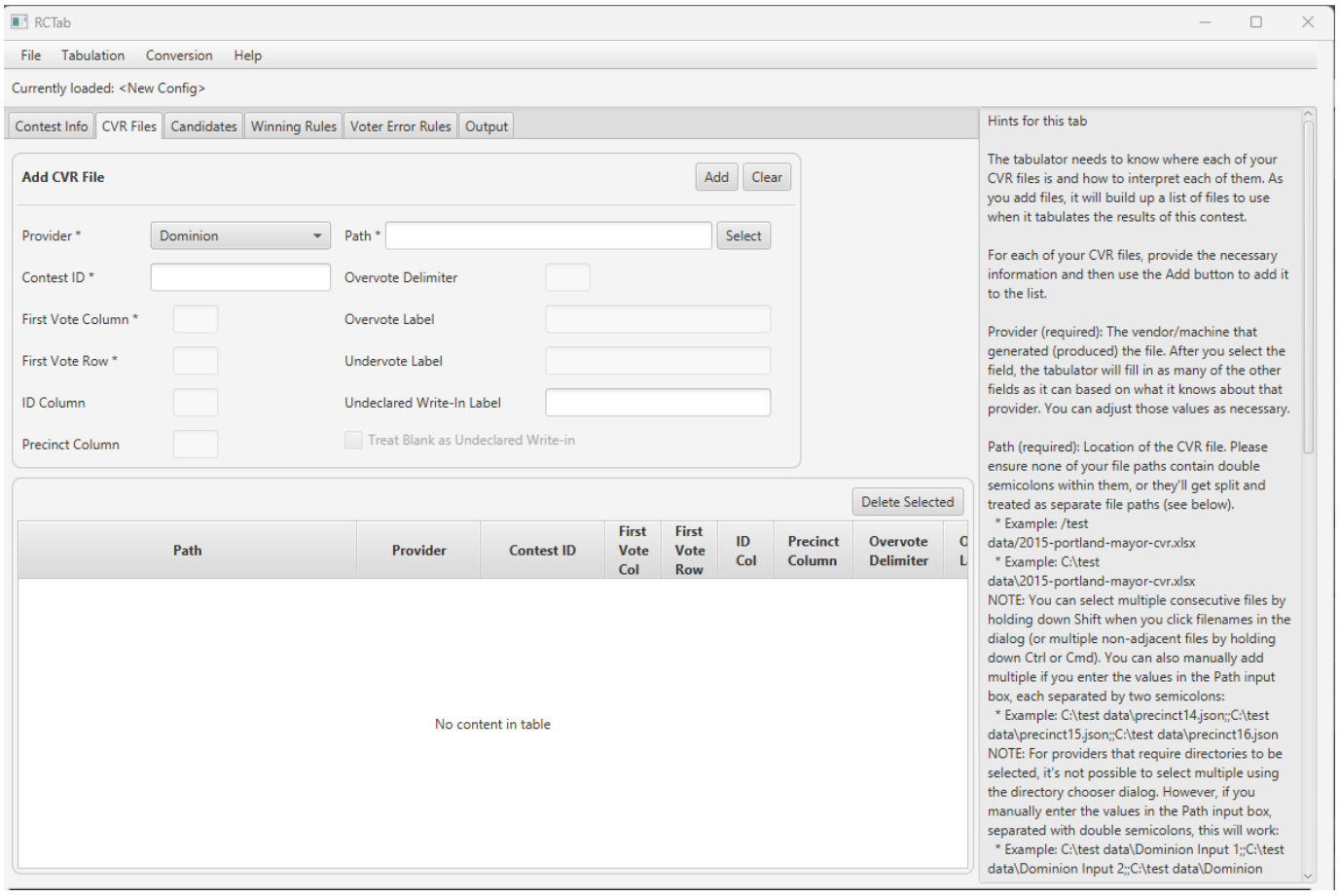
Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Overvote Label (Default value: overvote)	ID Column
Undeclared write-in label	Precinct Column
	Overvote Delimiter
	Undervote Label
	Treat Blank as Undeclared Write-In

CLEAR BALLOT



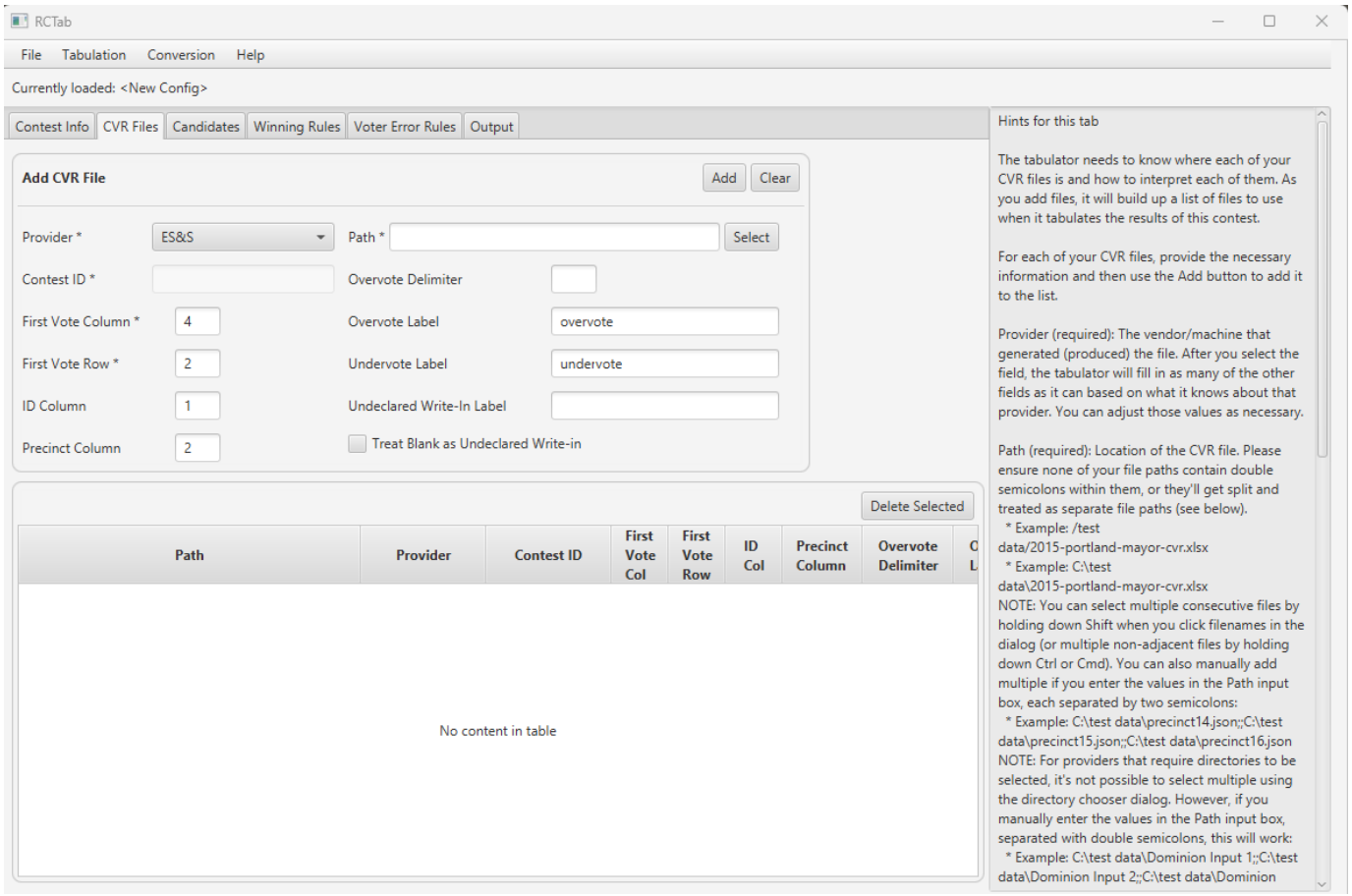
Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

DOMINION



Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

ES&S



Permits user to edit these options:	Does not permit user to edit these options:
Path	Contest ID
First Vote Column (Default Value: 4)	
First Vote Row (Default Value: 2)	
ID Column (Default Value: 1)	
Precinct Column (Default Value: 2)	
Overvote Delimiter	
Overvote Label (Default Value: overvote)	
Undervote Label (Default Value: undervote)	
Undeclared Write-In Label	
Treat Blank as Undeclared Write-In	

It is assumed that the ES&S export uses the default settings so below are the ES&S default settings. The user will only need to enter the path, but all other default settings are included. If the default settings are not used in the ES&S export file, the user must determine the values to be used. If defaults are not being used, users should confirm the proper values for First Vote Column, First Vote Row, ID Column, Precinct Column, Overvote Label, Undervote Label, and Undeclared Write-In Label by referencing the CVRs files to be used in the Tabulation.

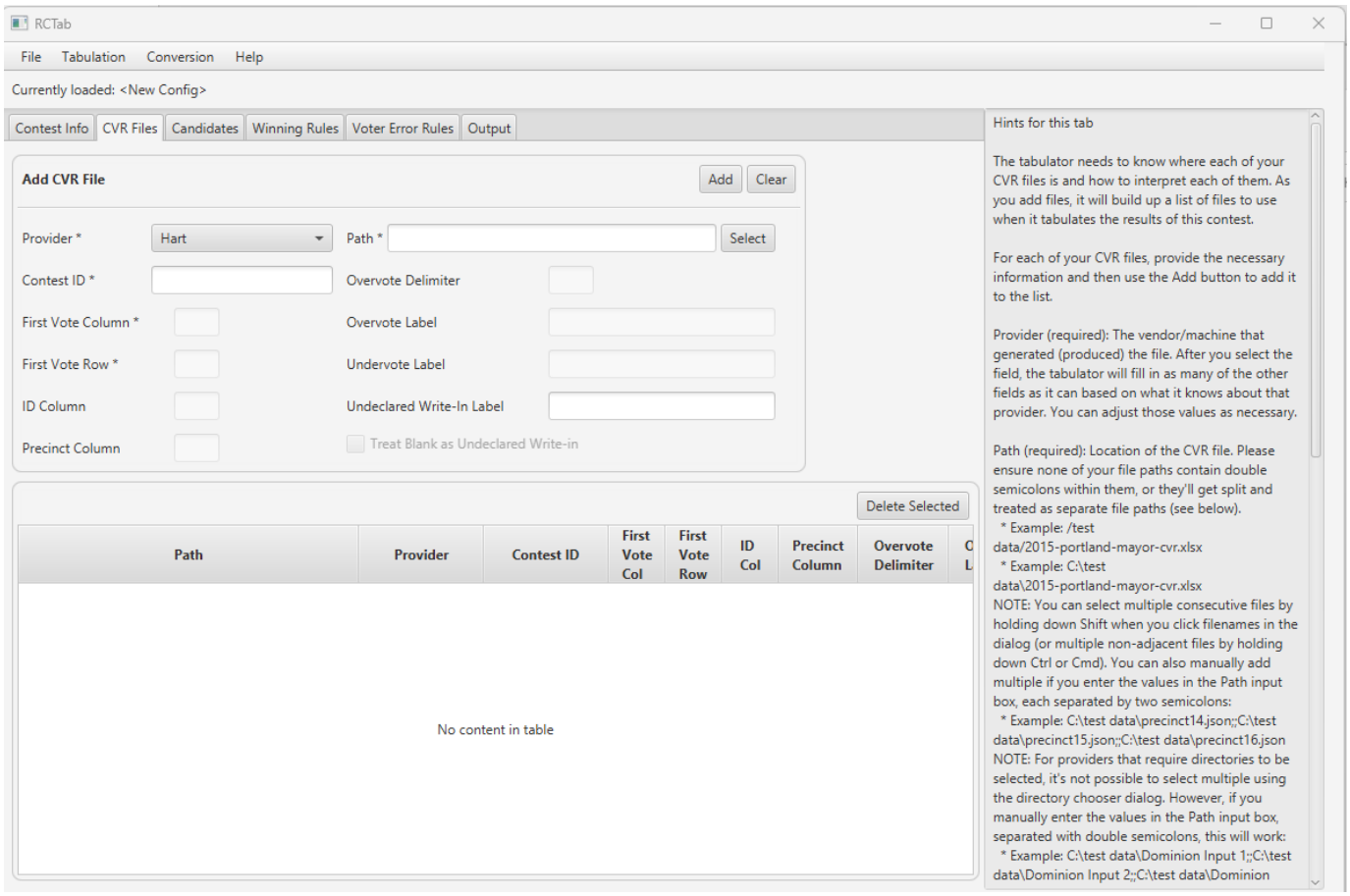
The First Vote Column location depends on the information users choose to export in their CVR exports from ES&S's system. ES&S systems can include individual CVR ID information, Precinct information, and ballot style label information in the first

three columns of a CVR, as shown in the below screenshot. Column A (aka Column 1) has CVR ID numbers; Column B (Column 2) has Precinct information, Column C (3) has ballot style information, and Column D (4) includes the first ranking in the RCV contest in this CVR. This is the standard CVR layout for ES&S. More information is available from ES&S.

A	B	C	D
Cast Vote Record	Precinct	Ballot Style	Rep. to Congress 1st Choice District 2
1	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
2	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
3	Fayette	CAN Ballot Style 130	DEM Golden, Jared F.
4	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
6	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
8	Fayette	CAN Ballot Style 130	Hoar, William R.S.
10	Fayette	CAN Ballot Style 130	undervote
15	Fayette	CAN Ballot Style 130	DEM Golden, Jared F.
17	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
19	Fayette	CAN Ballot Style 130	REP Poliquin, Bruce
20	Fayette	CAN Ballot Style 130	Hoar, William R.S.

CVR exports from ES&S contain columns for each ranking in an RCV contest. See the above screenshot for an example. This screenshot includes the five rankings voters had in the contest, arranged in sequential order. This CVR export included only data for the one RCV contest to be run. The setting “Maximum Number of Candidates That Can Be Ranked” (covered below in Winning Rules) instructs RCTab how many columns to process when running the round-by-round count. If that setting is set to five and the First Vote Column setting is set to four (Column D), RCTab will process this CVR starting at Column D/4 (column number 4) and continue through columns E/5, F/6, G/7, and H/8 - five columns for the five rankings voters have.

HART

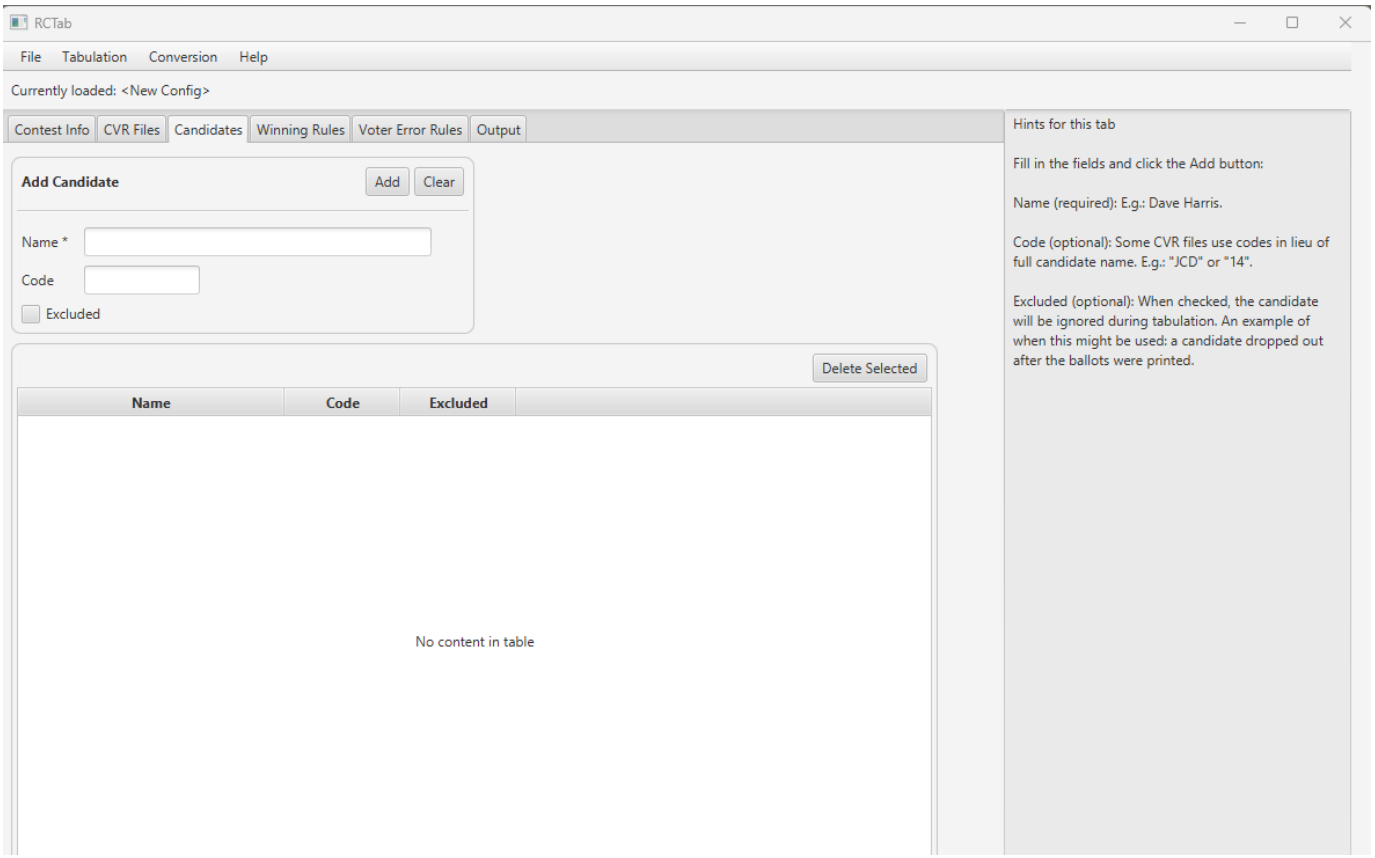


Permits user to edit these options:	Does not permit user to edit these options:
Path	First Vote Column
Contest ID	First Vote Row
Undeclared Write-In Label	ID Column
	Precinct Column
	Overvote Delimiter
	Overvote Label
	Undervote Label
	Treat Blank as Undeclared Write-In

Path: Select the path of the folder that contains signed Hart CVRs.

Candidates Tab

The candidates tab allows users to put in information about how candidates are referred to in cast-vote record files. These settings impact how CVR files are read. Candidate names entered on this tab will also be used to display candidate names in results files. Below is a screenshot of the Candidates Tab when a user first navigates to it.



Fill in the fields and click the Add button:

Name (required)**:** E.g.: Dave Harris. This information is used to display candidate names in results files.

Code (optional)**:** Some CVR files use codes in lieu of the full candidate name. e.g.: "JCD" or "14".

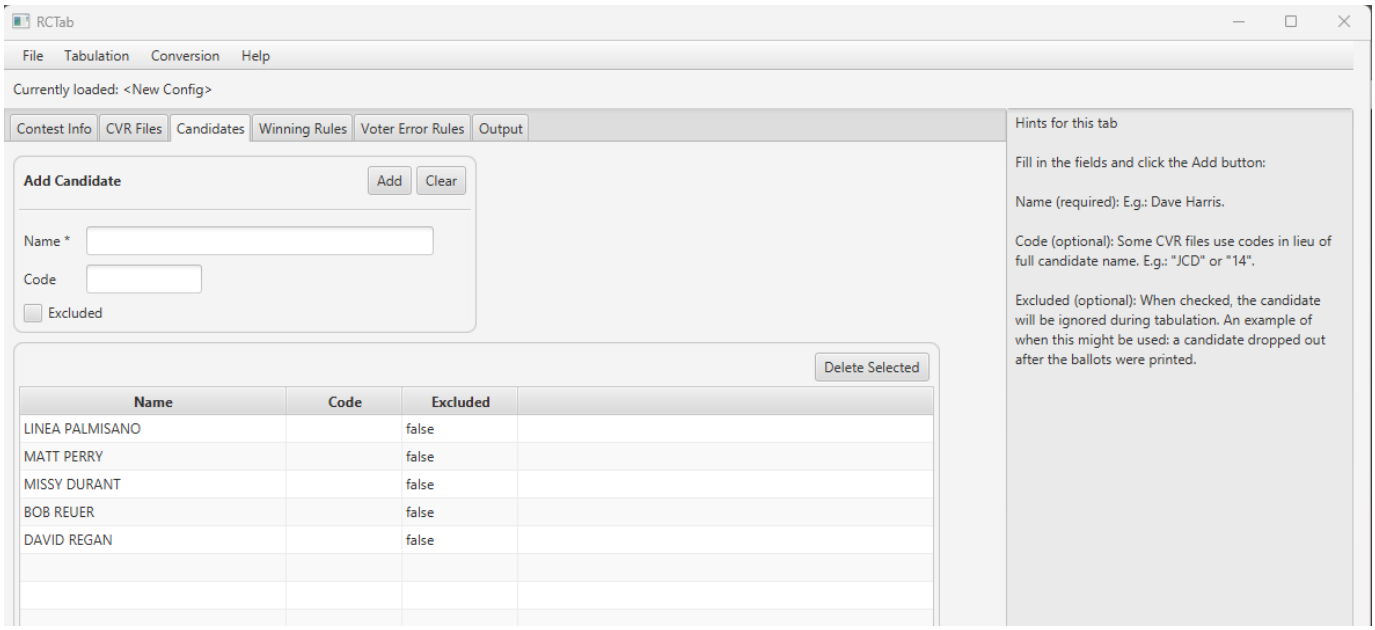
Excluded (optional)**:** When checked, the candidate will be ignored during tabulation. An example of when this might be used: a candidate dropped out after the ballots were printed.

Add: Adds Name, code, and/or excluded information to the Candidates Table.

Clear: Clears any information in Name, Code, and Excluded.

Delete Selected: Deletes selected candidate data from the Candidate Table.

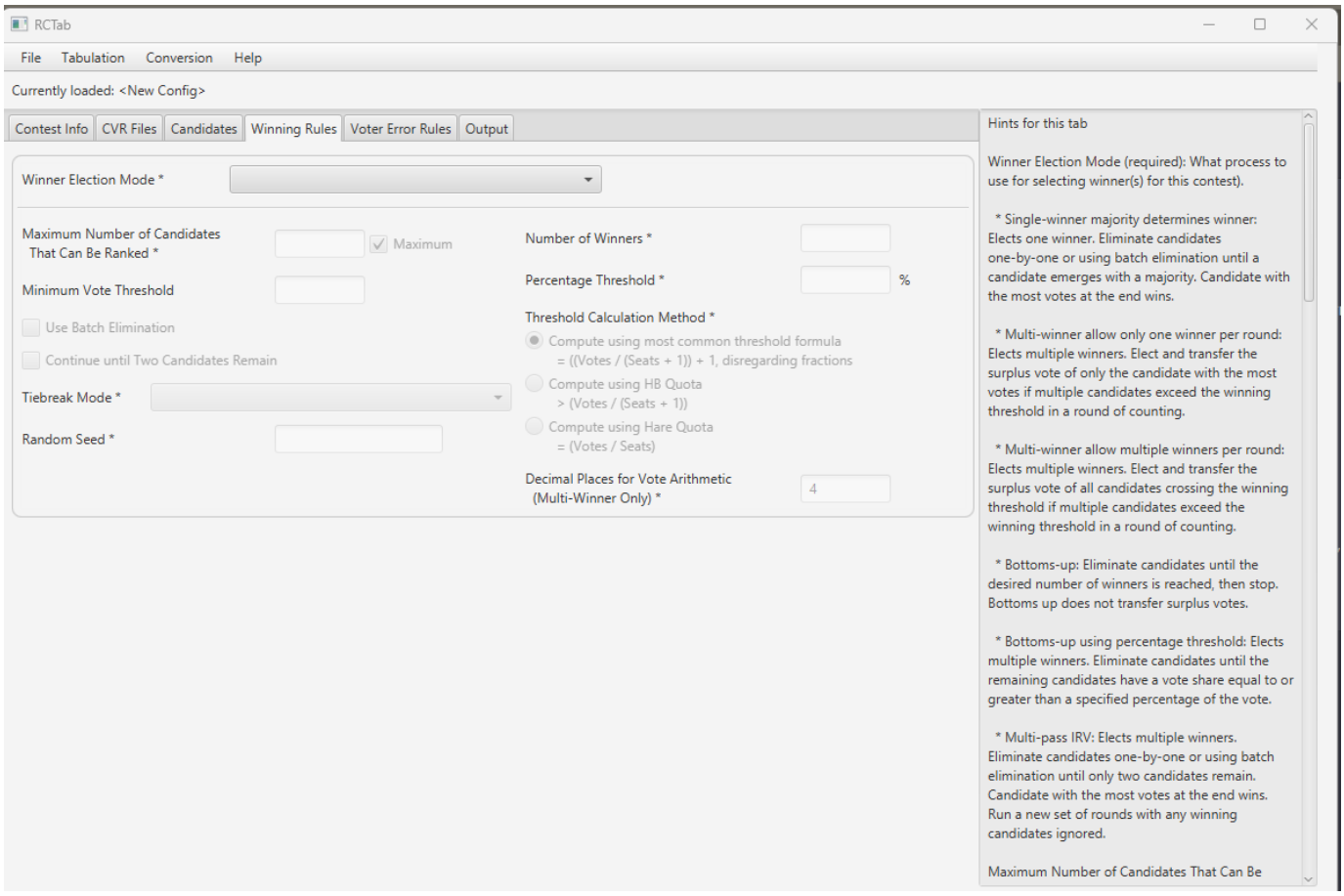
Below is a screenshot of an example of a Candidates Tab while being filled out:



Winning Rules Options

Winning Rules options tell RCTab what kind of ranked choice voting election to run and how to handle details required for each kind of ranked choice voting RCTab can run.

This is a screenshot of what winning rules look like when a user first navigates to them.



There are two main sets of options on the Winning Rules tab: Winner Election Mode options and Tiebreak Mode options.

Winner election mode options have many options that interact in many different ways. Which options are required and which options users can edit ultimately depend on the winner election mode option itself. Options are described below. Screenshots are then provided of each interface for each winner election mode.

Winner Election Mode (required)**:** What process to use for selecting winner(s) for this contest.

- **Single-winner majority determines winner:** Elects one winner. Eliminate candidates one-by-one or using batch elimination until a candidate emerges with a majority. Candidate with the most votes at the end wins.
- **Multi-winner allows only one winner per round:** Elects multiple winners. Elect and transfer the surplus vote of only the candidate with the most votes if multiple candidates exceed the winning threshold in a round of counting.
- **Multi-winner allows multiple winners per round:** Elects multiple winners. Elect and transfer the surplus vote of all candidates crossing the winning threshold if multiple candidates exceed the winning threshold in a round of counting.
- **Bottoms-up:** Eliminate candidates until the desired number of winners is reached, then stop. Bottoms up does not transfer surplus votes.
- **Bottoms-up using percentage threshold:** Elects multiple winners. Eliminate candidates until the remaining candidates have a vote share equal to or greater than a specified percentage of the vote.
- **Multi-pass IRV:** Elects multiple winners. Eliminate candidates one-by-one or using batch elimination until only two candidates remain. Candidate with the most votes at the end wins. Run a new set of rounds with any winning candidates ignored.

Maximum Number of Candidates That Can Be Ranked (required)**:** How many rankings each voter has in this contest. This tells RCTab how many rankings to read from each CVR. This option can be filled out in the text box or by selecting a checkbox. Selecting the checkbox disables the text box option and inserts the value “maximum”. The checkbox can be unselectedDefault value: Maximum. Available in all winner election modes.

Minimum Vote Threshold (optional)**:** The number of first-choice votes a candidate must receive in order to remain in the race. Any candidates falling below the minimum vote threshold are eliminated and have their votes transferred. Most jurisdictions do not set a minimum vote threshold. Note that if no candidate exceeds the minimum vote threshold, vote tabulation will silently fail. Be sure not to set a minimum vote threshold beyond the number of votes candidates have. This option can impact how any winner election mode tabulation runs. Its validity is not impacted by any other option. Available in all winner election modes.

Use Batch Elimination (optional)**:** Batch elimination, or simultaneous elimination of all candidates for whom it is mathematically impossible to be elected, eliminates all candidates who cannot receive enough votes to surpass the candidate with the next highest number of votes. Example: in a six candidate contest with 200 votes, Candidate A has 80 votes, Candidate B has 70, and the other four combined have 50. Because those four candidates can never combine their votes to surpass Candidate B, they can be batch eliminated. This option impacts how single-winner majority determines winner tabulation runs. Its validity is impacted by the winner election mode. It will not impact who wins the election. Available only when Winner Election Mode is "Single-winner majority determines winner".

Continue until Two Candidates Remain (optional)**:** Single-winner ranked choice voting elections can stop as soon as a candidate receives a majority of votes, even though 3 or more candidates may still be in the race. Selecting this option will run the round-by-round count until only two candidates remain, regardless of when a candidate wins a majority of votes. It will not impact who wins the election. Available only when Winner Election Mode is "Single-winner majority determines winner".

Number of Winners (required)**:** The number of seats to be filled in the contest. This option impacts the calculation of the election threshold. Its validity and editability is impacted by the winner election mode. It sets the value for “S” in the calculations described in the UI/configuration file parameters. Available and required for Multi-winner allow only one winner per round, Multi-winner allow multiple winners per round, Bottoms-up, and Multi-pass IRV. Set to 1 by RCTab for Single-winner majority determines winner.

- Note that multi-pass IRV does not use this value to calculate its election threshold. The election threshold in multi-pass IRV is always calculated using an S value of 1. This number is used in multi-pass IRV to determine how many instances of a single-winner majority determines winner tabulation must be run on CVR files. See Configuration Parameters for more details.

Percentage Threshold: The share of votes a candidate must have in order to win. Used to calculate the election threshold in “Bottoms-up using percentage threshold.” Candidates falling below this threshold are eliminated one-by-one beginning with the candidate with the fewest votes. Required and Available only when Winner Election Mode is "Bottoms-up using percentage threshold".

Threshold Calculation Method: The threshold of election is the number of votes a candidate must receive in order to win the election. There are three primary ways to calculate the threshold of election in multi-winner RCV contests. This will be set in law (either by statute or regulation) in your jurisdiction. Required and available only when Winner Election Mode is "Multi-winner allow only one winner per round" or "Multi-winner allow multiple winners per round".

- **Compute using most common threshold formula:** The most common threshold formula is calculated by dividing the number of votes by the number of seats plus one, then adding one to that number. Fractions are disregarded. This is also known as the Droop quota. Candidates must receive this number of votes (or more) to win. This is the default threshold calculation.
- **Compute using HB Quota**: The HB, or Hagenbach-Bischoff, Quota divides the number of votes by the number of seats plus one, leaving fractions. Candidates must receive more than this number of votes to win.
- **Compute using Hare Quota**: The Hare quota divides the number of votes by the number of seats. Fractions are disregarded. It requires candidates to receive that number of votes (or more) to win.

Decimal Places for Vote Arithmetic (Multi-Winner Only): Sets how many decimal places after the decimal point are used in surplus transfers and in calculating the threshold. Its validity and editability is impacted by the winner election mode. Required and available only when Winner Election Mode is "Multi-winner allow only one winner per round" or "Multi-winner allow multiple winners per round". Default value: 4.

TIEBREAKING

There are six tiebreak modes (described in technical detail in **Section 25 - Configuration File Parameters**). Tiebreak modes impact the validity/operation of one other option: random seed. Tiebreak mode is required for all winner election modes.

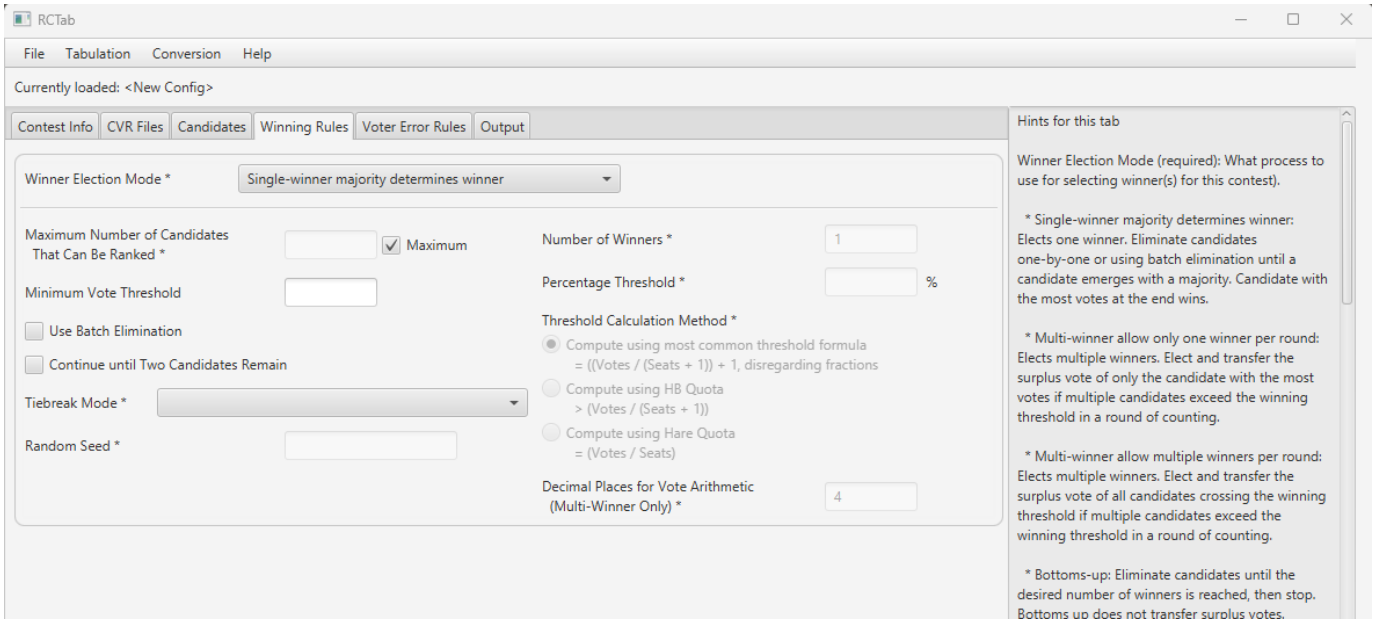
Tiebreak Mode (required for all winner election modes)**:** Ties in ranked choice voting contests can occur when eliminating candidates or when electing candidates. All winner election modes may have a tie for last place. Tiebreak mode breaks the tie and determines which candidate loses the tie. Multi-winner allows only one winner per round contests can have ties between candidates who have both crossed the threshold of election; in that case, ties are broken to determine whose surplus vote value transfers first. Tiebreak procedures are set in law, either in the ranked choice voting law used in your jurisdiction or in the elections code more generally. Select the option from this list that complies with the law and procedure in your jurisdiction.

- **Random:** Randomly select a tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect. Requires a random seed.
- **Stop counting and ask:** Pause count when a tie is reached. The user is prompted to select any tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect.
- **Previous round counts (then random):** The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, the tie is broken randomly. Requires a random seed.
- **Previous round counts (then stop counting and ask):** The tied candidate with the least votes in the previous round loses the tie. If there is a tie in the previous round, the user is prompted to select any tied candidate to eliminate or, in multi-winner allow only one winner per round contests only, elect.
- **Use candidate order in the config file:** Use the order of candidates in the config file to determine tiebreak results. Candidates lower in the list lose the tiebreaker.
- **Generate permutation:** Generate a randomly ordered list of candidates in the contest. Candidates lower in the permutation lose the tiebreaker. Requires a random seed.

Random Seed (required if Tiebreak Mode is "Random", "Previous round counts (then random)", or "Generate permutation"): Enter a positive or negative integer to generate random orders. Impacts operation of randomized functions for tiebreaking. Unavailable if Tiebreak Mode is Stop Counting and Ask, Previous round counts (then stop counting and ask) or Use candidate order in the config file.

Following are screenshots of each of the winning rules interfaces on RCTab.

SINGLE-WINNER MAJORITY DETERMINES WINNER



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Use Batch Elimination

Continue until Two Candidates Remain

Tiebreak Mode*

Random Seed (* if using randomized)

Automatically fills in and locks this option:

Number of Winners*

Invalid options, locked for this mode:

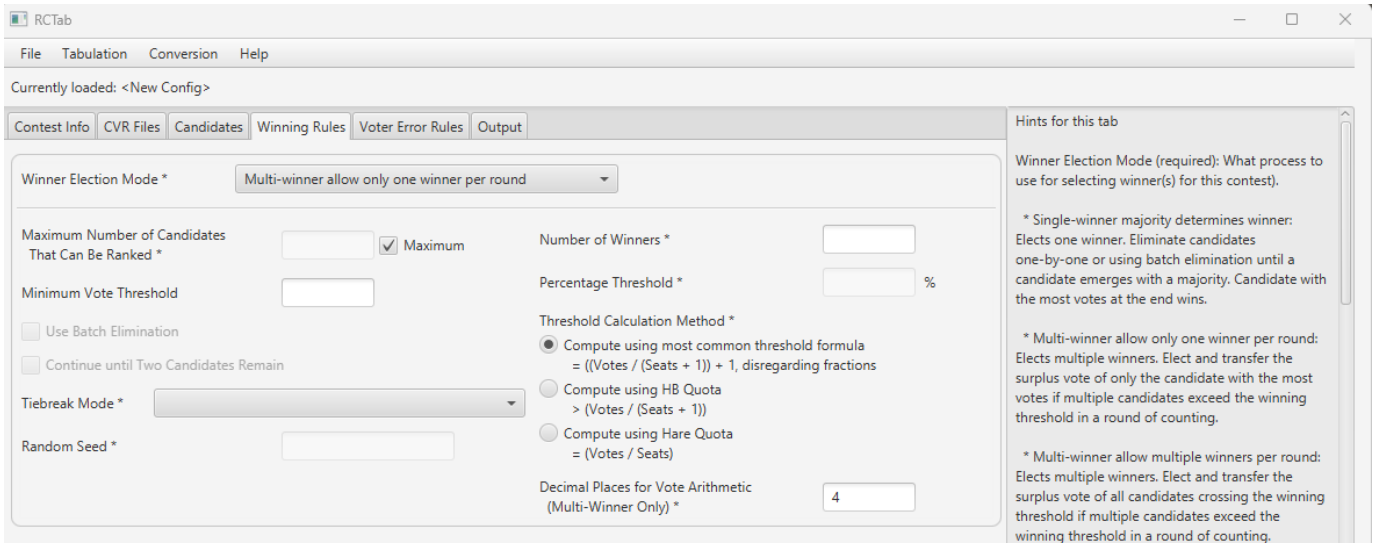
Percentage Threshold

Threshold Calculation Method

Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

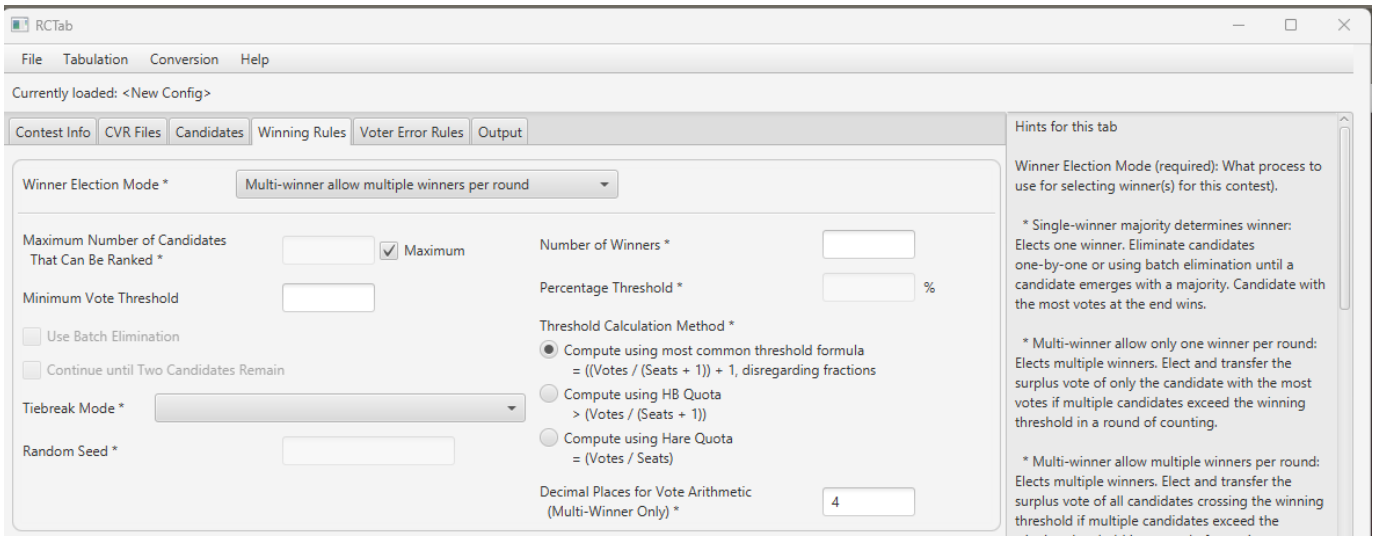
MULTI-WINNER ALLOW ONLY ONE WINNER PER ROUND



Permits user to edit these options:	**Invalid options, locked for this mode:**
Maximum Number of Candidates That Can Be Ranked*	Percentage Threshold
Minimum Vote Threshold	Use Batch Elimination
Tiebreak Mode*	Continue until Two Candidates Remain
Random Seed (* if using randomized tiebreak mode)	
Number of Winners*	
Threshold Calculation Method*	
Decimal Places for Vote Arithmetic*	

Note: Required settings are denoted with an asterisk (*).

MULTI-WINNER ALLOW MULTIPLE WINNERS PER ROUND



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Tiebreak Mode*

Random Seed (* if using randomized tiebreak mode)

Number of Winners*

Threshold Calculation Method*

Decimal Places for Vote Arithmetic*

Invalid options, locked for this mode:

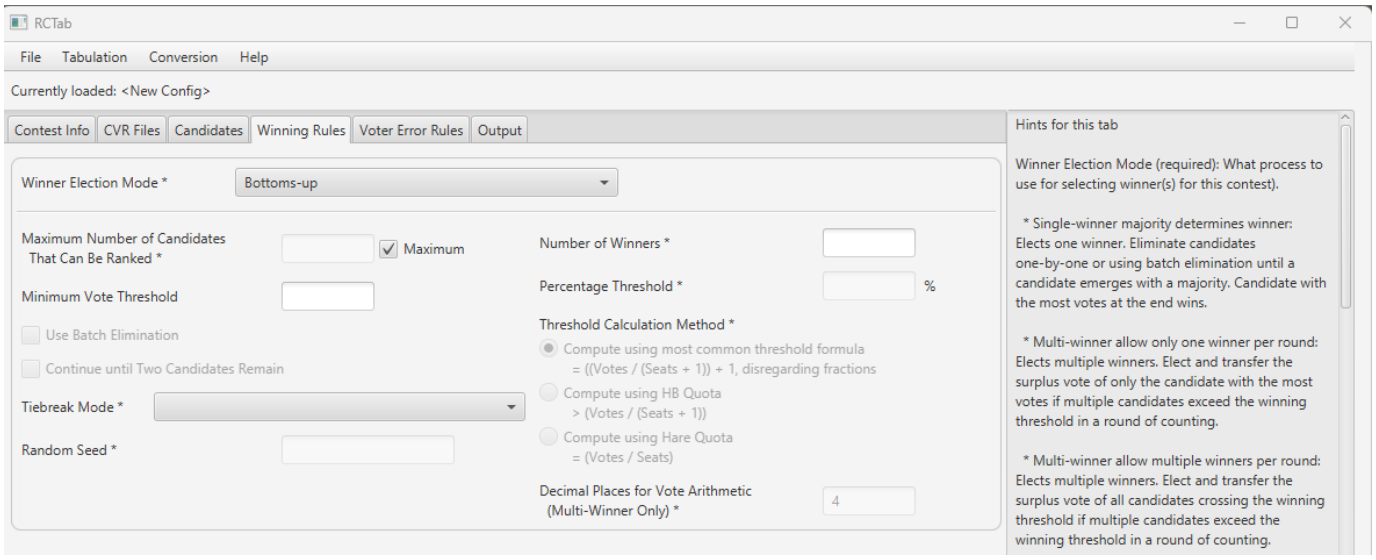
Use Batch Elimination

Continue until Two Candidates Remain

Percentage Threshold

Note: Required settings are denoted with an asterisk (*).

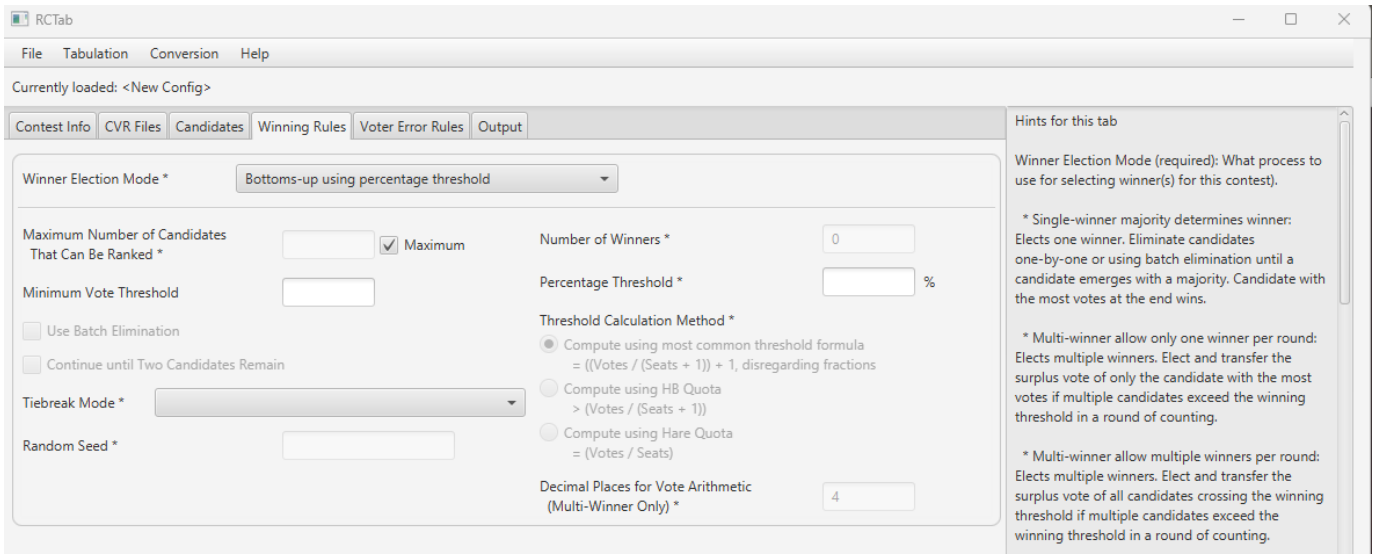
BOTTOMS-UP



Permits user to edit these options:	**Invalid options, locked for this mode:**
Maximum Number of Candidates That Can Be Ranked*	Use Batch Elimination
Minimum Vote Threshold	Continue until Two Candidates Remain
Tiebreak Mode*	Percentage Threshold
Random Seed (* if using randomized tiebreak mode)	Threshold Calculation Method
Number of Winners*	Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

BOTTOMS-UP USING PERCENTAGE THRESHOLD



Permits user to edit these options:

Maximum Number of Candidates That Can Be Ranked*

Minimum Vote Threshold

Tiebreak Mode*

Random Seed (* if using randomized tiebreak mode)

Percentage Threshold*

Automatically fills in and locks this option:

Number of Winners*

Invalid options, locked for this mode:

Use Batch Elimination

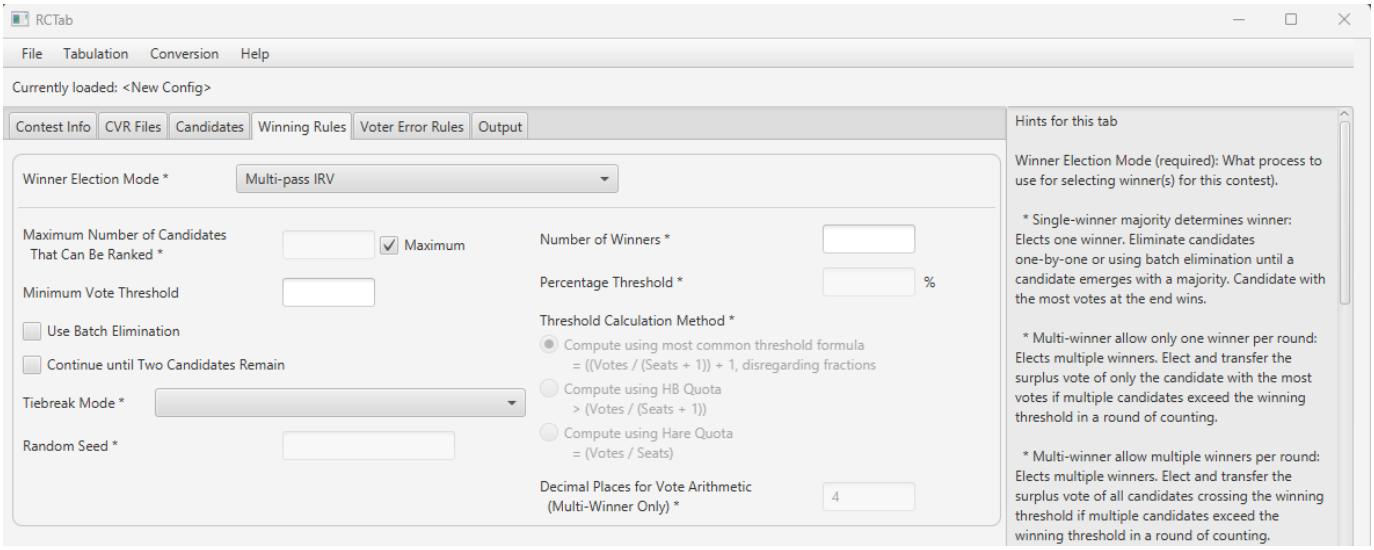
Continue until Two Candidates Remain

Threshold Calculation Method

Decimal Places for Vote Arithmetic

Note: Required settings are denoted with an asterisk (*).

MULTI-PASS IRV

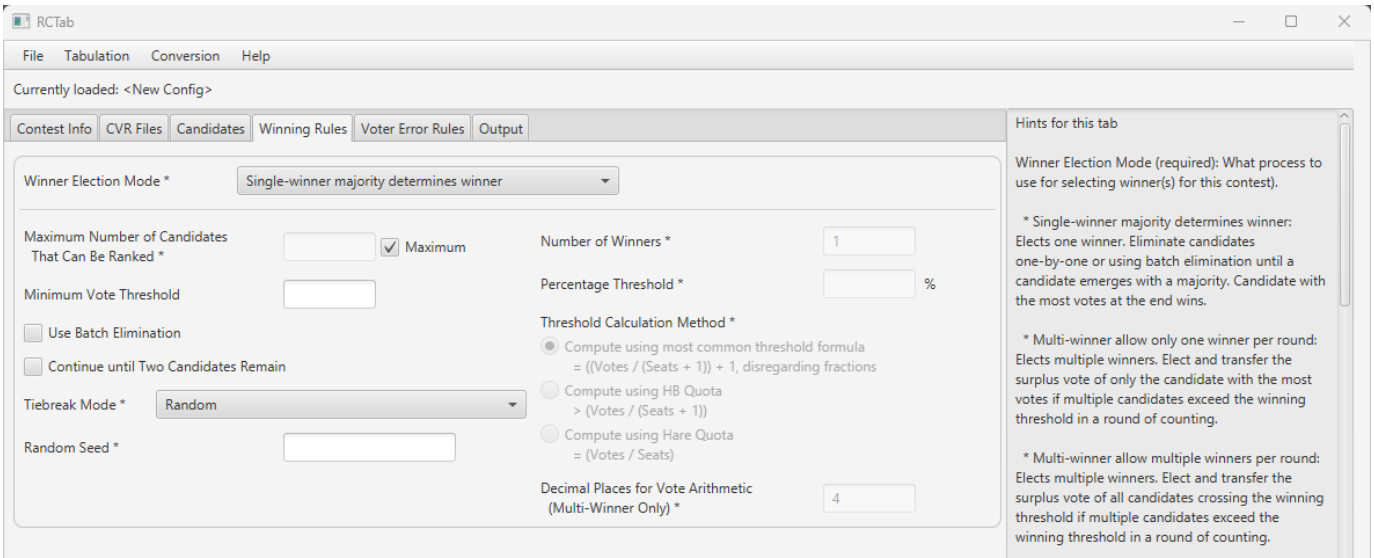


Permits user to edit these options (required settings denoted with *):	Invalid options, locked for this mode:
Maximum Number of Candidates That Can Be Ranked*	Use Batch Elimination
Minimum Vote Threshold	Continue until Two Candidates Remain
Tiebreak Mode*	Threshold Calculation Method
Random Seed (* if using randomized tiebreak mode)	Decimal Places for Vote Arithmetic
Number of Winners*	Percentage Threshold

TIEBREAK MODES

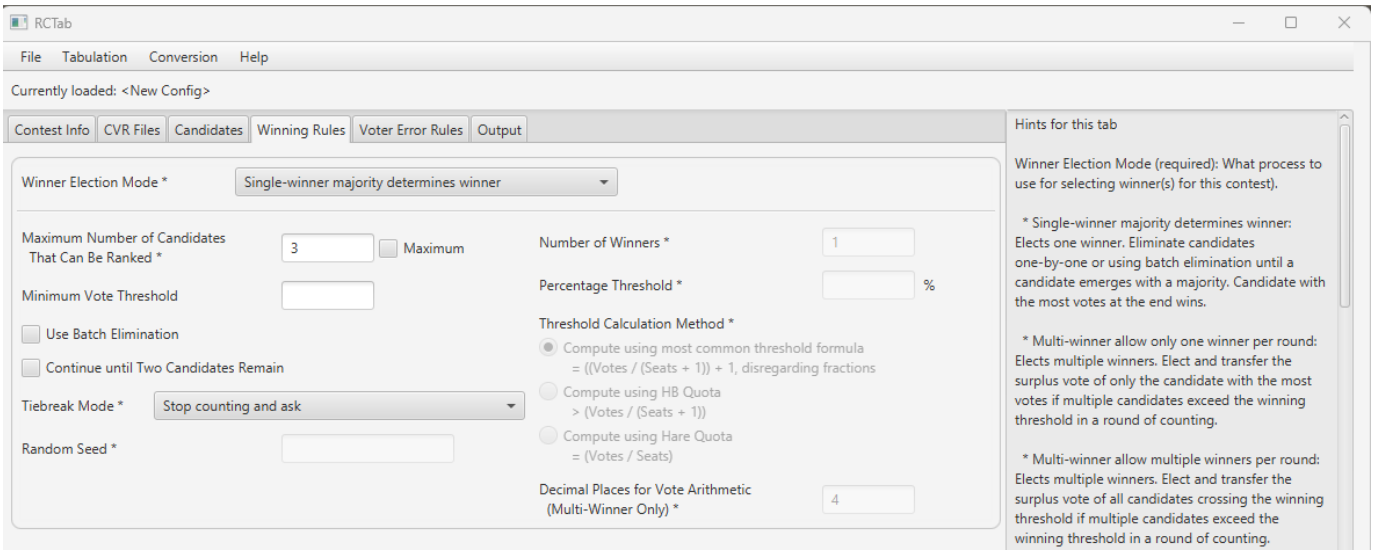
Random, Generate Permutation, Previous Round Counts (then random)

If users select any of the tiebreaking modes that incorporate randomness (random, generate permutation, previous round counts (then random)) this is how RCTab will appear. Users must fill out the Random Seed setting.



Stop counting and ask, previous round counts (then stop counting and ask), use candidate order in the config file

If users select any of the tiebreaking modes that incorporate user input (Stop counting and ask, previous round counts (then stop counting and ask), use candidate order in the config file) this is how RCTab will appear. The Random Seed field will be locked.



Voter Error Rules

The tabulator needs to know how to handle voter errors in your jurisdiction. These requirements are typically included in statute or regulation. This is a screenshot of what winning rules looks like when a user first navigates to it.

Overvote Rule (required)**:** How to handle a ballot where a voter has marked multiple candidates at the same ranking when that ballot is encountered in the round-by-round count.

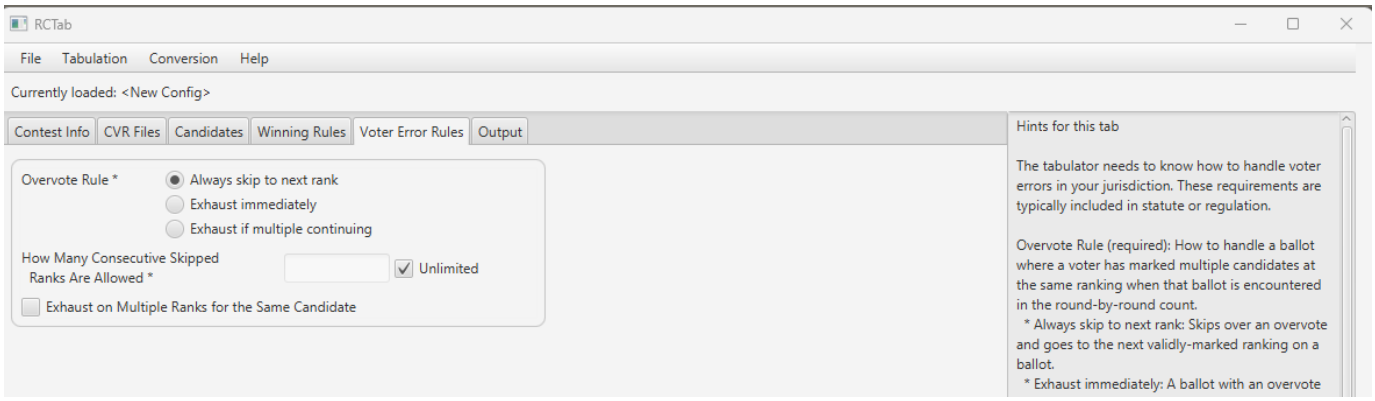
- **Always skip to next rank:** Skips over an overvote and goes to the next validly-marked ranking on a ballot.
- Requires Overvote Label to be supplied (when overvote label can be edited)
- **Exhaust immediately:** A ballot with an overvote exhausts when that overvote is encountered in the rounds of counting.
- Requires overvote label to be supplied (when overvote label can be edited)
- **Exhaust if multiple continuing:** If a voter has an overvote but only one candidate at that overvote is still in the race when that overvote is encountered, the ballot counts for that candidate. If multiple candidates at the overvote are still in the race, the ballot exhausts.
- Requires overvote delimiter to be supplied (when overvote delimiter can be edited)

How Many Consecutive Skipped Ranks Are Allowed (required)**:** How many rankings in a row can a voter skip and still have later rankings count? 0 allows no skipped rankings. `1` allows voters to skip rankings one at a time, but not more than `1` in a row, and so on. Selecting the checkbox disables the text box and inserts the value “unlimited”. Checkbox can be unselected. Default value: `1`. Example: A voter could rank their 1st, 3rd and 5th choices and not exhaust the ballot under this rule, for example.

Exhaust on Multiple Ranks for the Same Candidate (optional)**:** When checked, the tabulator will exhaust a ballot that includes multiple rankings for the same candidate when that repeat ranking is reached. When unchecked repeated rankings will not exhaust the ballot. Example: A voter ranks the same candidate 1st and 3rd, a different candidate 2nd, and another candidate

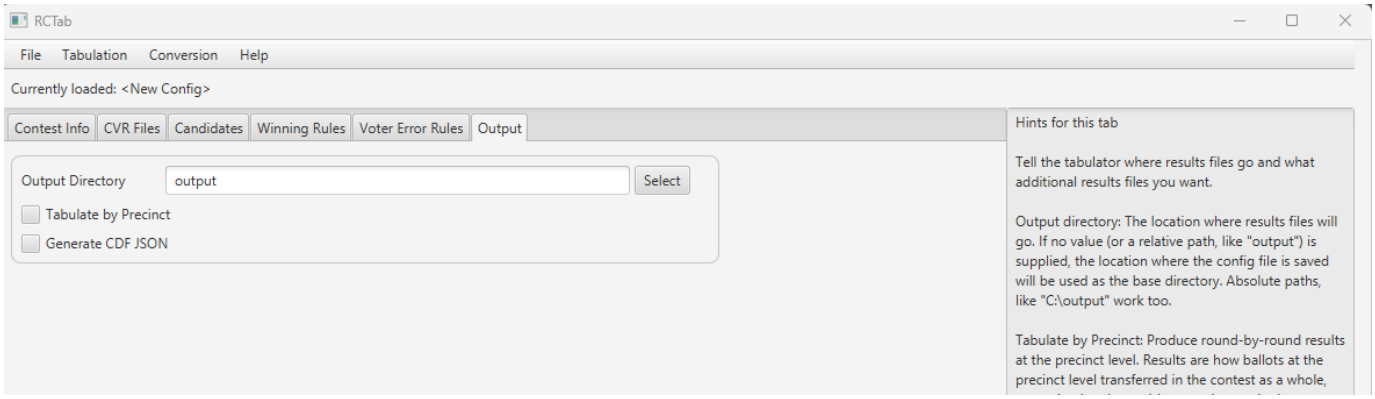
4th. If their original first choice and their second choice are eliminated, the ballot exhausts when it reaches the repeat ranking in rank 3. The ranking in the 4th rank does not count.

Below is an example screenshot of a sample filled-out voter error rules tab. The user has selected Always skip to next rank for Overvote, has selected unlimited for How Many Consecutive Skipped Ranks Are Allowed (making the text box inoperable) and has left Exhaust on Multiple Ranks for the Same Candidate blank.



Output Tab

Tell the tabulator where results files go and what additional results files you want. This is a screenshot of what output looks like when a user first navigates to it.



Output directory: One folder per contest tabulated should be created. Output directory instructs RCTab where to save any output files from successful RCTab tabulations. The location where results files will go. If no value (or a relative path, like `output`) is supplied, the location where the config file is saved will be used as the base directory. Absolute paths, like `C:\output` work too. Select allows users to navigate through Explorer to select a location.

RCTab will not allow the output path to be configured to a Windows user account folder (by default anything under the path `c:\users\`). This requirement ensures read-only output. You **must** follow the instructions in **Section 22 - Installation Instructions for Windows OS** to set permissions on the output folder. Following these instructions ensures that the 'RCTab' user account cannot edit or delete RCTab summary output files or audit logs.

Tabulate by Precinct: Produce round-by-round results at the precinct level. Results are how ballots at the precinct level transferred in the contest as a whole, not a simulated round-by-round count in the precinct. Requires precinct information in CVR Files tab for ES&S CVRs. Impacts which files are written to the output directory.

Generate a CDF JSON: Produce a VVSG common data format JSON file of the CVR. Impacts which files are written to the output directory.

3.1.4 Generating Results Files

Validating Configuration files

Once a configuration file is successfully created, the user must validate the configuration file.

1. Click on "Tabulation" at the top of the software window
2. Click on "Validate"
3. 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.
4. If any error messages appear in the log box, refer to **Section 29 - RCTab Operator Log Messages** and messages in the log box for how to resolve errors. If the error persists, restart the RCTab software

Saving Configuration Files

Once ready to run a tabulation, the user must first save the configuration file.

1. Click on "File" at the top of the software window
2. Click on "Save..."
3. Select a location to save the configuration file. The manufacturer suggests users save the configuration file to the same location set in the Output Directory setting.
4. 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.
5. If any error messages appear in the log box, refer to 430 RCTab Operator Log Messages documentation and messages in the log box for how to resolve errors. If the error persists, restart RCTab software.

Running a Tabulation

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

1. Click on "Tabulation" at the top of the software window
2. Click on "Tabulate"
3. Tabulation will begin.
4. If all the above steps were successfully completed, Tabulation will run until complete.
5. Tabulator log box will update with messages as Tabulation proceeds.
6. Once complete, the Tabulator log box will display a message stating `Results written to: [filepath from Output Directory]`

Output files will be:

- `.csv` contest summary files
- `summary.csv` Whole-contest summary file
- `summary.csv.hash` Corresponding hash file. This contains the hash to verify the results in `summary.csv`
- Precinct-by-precinct summary files (if tabulating by precinct)
- `.json` contest summary files
- `summary.json` Whole-contest summary file
- `summary.json.hash` Corresponding hash file. This contains the hash to verify the results in `summary.json`
- Precinct-by-precinct summary files (if tabulating by precinct)
- `.log` audit files
- `.log` audit files are exported in 50MB sections. If a `.log` file exceeds 50MB an additional `.log` file is started by RCTab
- Corresponding `audit_N.log.hash` file. This contains the hash to verify the information in each `audit_N.log file`
- `.json` CDF (common data format) files if Generate a CDF JSON is checked

If necessary, instructions for verifying result file hashes can be found in **Section 23 - Trusted Build & Output Hash Verification - Windows OS**.

Users can then navigate to "File" and click "Exit" if all contests are tabulated.

If more contests remain to be tabulated, and contests will contain fewer than 1,000,000 total votes, user can navigate to “File” and click “New.” This will clear all fields in RCTab and permit the user to create a new configuration file.

If contests to be tabulated will contain more than 1,000,000 total votes, return to start of guide and re-launch tabulator according to large configuration launch requirements. Then follow the guide to set up a configuration file.

If any errors arise in the use of RCTab, refer to **Section 29 - RCTab Operator Log Messages**. Errors arising out of any hardware or software other than RCTab should refer to **Section 09 - System Maintenance Manual** and any relevant user and maintenance manuals.

Before publishing results, jurisdictions should use their established reconciliation procedures to ensure total votes counted in each round equals total ballots cast in the contest. If numbers in the reconciliation process do not match, the user should double-check that all CVRs for that contest were exported successfully from the voting system and run RCTab process for that contest again. Rely on user jurisdiction CVR handling procedures for transmitting CVRs.

Any interaction with RCTab, including producing configuration files, running tabulations, hashing results files, and transmission of files from RCTab on USB drives should follow transmission procedures required in the jurisdiction, including the use of a team of no less than two trained personnel.

Required capabilities that may be bypassed or deactivated during installation or operation by the user shall be clearly indicated. Additional capabilities that function only when activated during installation or operation by the user shall be clearly indicated. Additional capabilities that normally are active but may be bypassed or deactivated during installation or operation by the user shall be clearly indicated.

The installation process for RCTab software does not give users the opportunity to bypass or deactivate options or settings.

Capabilities that are active or inactive in software operation depend on various factors. Many of these factors are laid out above. Configuration files determine which system capabilities apply to a given set of voting data. More information about operations that users can set through the user interface is provided in **Section 25 - Configuration File Parameters**. Information about the operation of those settings is also provided in **Section 02 - Software Design and Specifications.**

3.1.5 Configuration File Parameters and UI Label Match Sheet

Below is a list of all configuration file parameter labels and their corresponding labels in RCTab UI. Labels are organized by the order of their appearance in the RCTab UI.

Configuration File Parameters Name/Label	UI Name/Label
`contestName`	Contest Name
`contestDate`	Contest Date
`contestJurisdiction`	Contest Jurisdiction
`contestOffice`	Contest Office
`rulesDescription`	Rules Description
`provider`	Provider
`cdf`	CDF
`clearBallot`	Clear Ballot
`Dominion`	Dominion
`ess`	ES&S
`hart`	Hart
`filePath`	Path
`contestId`	Contest ID
`firstVoteColumnIndex`	First Vote Column
`firstVoteRowIndex`	First Vote Row
`idColumnIndex`	ID Column
`precinctColumnIndex`	Precinct Column
`overvoteDelimiter`	Overvote Delimiter
`overvoteLabel`	Overvote Label
`undervoteLabel`	Undervote Label
`undeclaredWriteInLabel`	Undeclared Write-in Label
`treatBlankAsUndeclaredWriteIn`	Treat Blank as Undeclared Write-In
`name`	Name
`code`	Code
`excluded`	Excluded
`winnerElectionMode`	Winner Election Mode
`singleWinnerMajority`	Single-Winner Majority Determines Winner
`multiWinnerAllowOnlyOneWinnerPerRound`	Multi-Winner Allow Only One Winner Per Round
`multiWinnerAllowMultipleWinnersPerRound`	Multi-Winner Allow Multiple Winners Per Round
`bottomsUp`	Bottoms-up
`bottomsUpUsingPercentageThreshold`	Bottoms-up using Percentage Threshold
`multiPassIrv`	Multi-Pass IRV
`maxRankingsAllowed`	Maximum Number of Candidates That Can Be Ranked
`minimumVoteThreshold`	Minimum Vote Threshold

Configuration File Parameters Name/Label	UI Name/Label
`batchElimination`	Use Batch Elimination
`continueUntilTwoCandidatesRemain`	Continue Until Two Candidates Remain
`tiebreakMode`	Tiebreak Mode
`random`	Random
`stopCountingAndAsk`	Stop counting and ask
`previousRoundCountsThenRandom`	Previous Round Counts (then random)
`prevousRoundCountsThenAsk`	Previous Round Counts (then stop counting and ask)
`useCandidateOrder`	Use candidate order in config file
`generatePermutation`	Generate permutation
`randomSeed`	Random Seed
`numberOfWinners`	Number of Winners
`multiSeatBottomsUpPercentageThreshold`	Percentage Threshold
`nonIntegerWinningThreshold`	Compute using HB Quota > (Votes / (Seats + 1))
`hareQuota`	Compute using Hare Quota = (Votes/Seats)
`decimalPlacesForVoteArithmetic`	Decimal Places for Vote Arithmetic (Multi-Winner Only)
`overvoteRule`	Overvote Rule
`alwaysSkipToNextRank`	Always skip to next rank
`ExhaustImmediately`	Exhaust Immediately
`exhaustIfMultipleContinuing`	Exhaust if multiple continuing
`maxSkippedRanksAllowed`	How Many Consecutive Skipped Ranks Are Allowed
`exhaustOnDuplicateCandidate`	Exhaust on Multiple Ranks for the Same Candidate
`outputDirectory`	Output Directory
`tabulateByPrecinct`	Tabulate by Precinct
`generateCdfJson`	Generate CDF JSON