Configuration File Parameters

URCVT v.1.2.0 300-NY Configuration File Parameters v.1.0.0 document is solely for use in the State of New York. This document can be expanded or updated as is necessary or required. Any recommendations listed in this document should not supersede user jurisdiction procedures or other controlling governance entities.

URCVT v.1.2.0 300-NY Configuration File Parameters v.1.0.0

This document lists the parameters the user will configure within the Universal RCV Tabulator. 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, URCVT will alert the user to the missing requirement and provide suggested resolution steps.

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.

Note: A previous version of this document is included in the URCVT software files. This document includes incorrect information related to file lengths (file lengths can only be 255 characters in Windows 10).

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:

1. "contestName" required
   a. The name of the contest

RANKED CHOICE VOTING RESOURCE CENTER | ELECTION ADMINISTRATION RESOURCE CENTER

 b. Used for naming audit output files
 c. Example: "Portland Mayoral Race 2017"
 d. Value: text string of length [1..255]
2. The "outputDirectory" optional
 a. Directory for audit output files (absolute or relative path)
 b. Example: /Path/To/TabulatorResults
 c. Example: output data/contest1
 d. Value: string of length [1..255]
 e. If not supplied: files will be saved to the current working directory
3. The "contestDate" optional
 a. Date of the contest
 b. Example: "2015-11-03"
 c. Value: text string of length [1..1000]
 d. If not supplied: none
4. The "contestJurisdiction" optional
 a. Text description of the jurisdiction of this contest
 b. Example: "Portland, ME"
 c. Value: text string of length [1..1000]
 d. If not supplied: none
5. "contestOffice" optional
 a. text description of the office being contested
 b. Example: "Mayor"
 c. Value: text string of length [1..1000]
 d. 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:

1. "provider" required
 a. text description of the vendor / machine which generated this file
 b. value: "cdf" | "clearBallot" | "dominion" | "ess" | "hart"
2. "filePath" required
 a. 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)
 b. example: /Users/test_data/2015-portland-mayor-cvr.xlsx
 c. value: string of length [1..255]
3. "contestId" required when CVR source files are from a provider other than ES&S; must be blank for ES&S
 a. the ID of the contest to tabulate, as represented in the CVR file(s)
 b. example: "b651b997-417a-46d9-a676-a43d4df94ddc"
 c. value: text string of length [1..1000]

4. "firstVoteColumnIndex" required and used if and only if provider is ES&S
    a. index of the column (starting from 1) that contains the top-ranked candidate for each CVR
    b. example: 3
    c. value: [1..1000]
5. "firstVoteRowIndex" required and used if and only if provider is ES&S
    a. index of the row (starting from 1) that contains the rankings for the first CVR
    b. example: 2
    c. value: [1..100000]
6. "idColumnIndex" optional and can be used only if provider is ES&S
    a. index of the column (starting from 1) that contains the unique ID for each CVR
    b. example: 1
    c. value: [1..1000]
7. "precinctColumnIndex" required and used if and only if "tabulateByPrecinct" is enabled and provider is ES&S
    a. index of the column (starting from 1) that contains the precinct name for each CVR
    b. example: 2
    c. value: [1..1000]
8. "overvoteLabel" optional and can be used only if provider is ES&S or CDF
    a. 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)
    b. example: "OVERVOTE"
    c. value: string of length [1..1000]
9. "undervoteLabel" optional and can be used only if provider is ES&S
    a. the special label used in the cast vote records to denote an undervote
    b. example: "UNDERVOTE"
    c. value: string of length [1..1000]
10. "undeclaredWriteInLabel" optional
    a. the special label used in the cast vote records to denote a vote for an undeclared write-in
    b. example: "UWI"
    c. value: string of length [1..1000]
11. "treatBlankAsUndeclaredWriteIn" optional and can be used only if provider is ES&S

a. tabulator will interpret a blank cell in a CVR as a vote for an undeclared write-in
b. value: true | false
c. 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:

1. "name" required
    a. Full name of the registered candidate
    b. Example: "Duson, Jill C."
    c. Value: string of length [1..1000]
2. "code" optional
    a. Candidate code which may appear in CVRs in lieu of full candidate name
    b. Example: "JCD"
    c. Value: string of length [1..1000]
    d. If not supplied: none
3. "excluded" optional
    a. Candidate should be ignored during tabulation
    b. Value: true | false
    c. If not supplied: false

"rules" required

Set of configuration parameters that specify the tabulation rules

The "rules" section contains the following parameters:

1. "tiebreakMode" required
    a. how the program should decide which candidate to eliminate when multiple candidates are tied for last place
    b. or which candidate to elect first when:
        i. 1) electionWinnerMode is set to multiWinnerAllowOnlyOneWinnerPerRound, and
        ii. 2) multiple candidates exceed the winning threshold in the same round, and
        iii. 3) at least two of those candidates are tied for the highest vote total in that round
    c. value: "random" | "stopCountingAndAsk" | "previousRoundCountsThenRandom" | "previousRoundCountsThenAsk" | "useCandidateOrder" | "generatePermutation"
        i. we use java.util.random for randomness in our tiebreak implementations

1. see: https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/Random.html
2. 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:

1. Random = new Random(config.getRandomSeed());

d. "Random"

i. during tabulation, in the event of a tie at the end of a round:

1. the list of tied candidates is sorted alphabetically
2. a randomDouble is generated using the random object:
3. double randomDouble = random.nextDouble();
4. the randomDouble is mapped to one of the tied candidates in the list:
5. int randomCandidateIndex = (int) Math.floor(randomDouble * (double) tiedCandidates.size());
6. the selected candidate will be the winner or loser for that round

e. "stopCountingAndAsk":

i. the user is presented with a list of tied candidates

ii. the user will input their selection manually

f. "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"

g. "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"

h. "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.
        i. "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

2. "overvoteRule" required
    a. how the program should handle an overvote when it encounters one
    b. value: "alwaysSkipToNextRank" | "exhaustImmediately" | "exhaustIfMultipleContinuing"
    c. Only “exhaustImmediately” is in scope
        i. "exhaustImmediately": exhaust a ballot as soon as we encounter an overvote
3. "winnerElectionMode" required
    a. which process the program should apply for selecting the winner(s)
    b. value: "singleWinnerMajority" | "multiWinnerAllowOnlyOneWinnerPerRound" | "multiWinnerAllowMultipleWinnersPerRound" | "bottomsUp" | "bottomsUpUsingPercentageThreshold" | "multiPassIrv"
    c. Only “singleWinnerMajority” is in scope. All other details listed below in Out of Scope details.
        i. "singleWinnerMajority": no special process (only valid when numberOfWinners = 1).
            1. Election threshold = floor(V/(S+1)) + 1
            2. where V = total number of votes (in the current round); and S = numberOfWinners
4. "numberOfWinners" required
    a. the number of seats to be won in this contest

   b. Uneditable if using “single-winner majority determines winner,” automatically set to 1. Uneditable if using “Bottoms-up using percentage threshold,” automatically set to 0.
   c. note: we use fractional vote transfer to redistribute votes in “multiWinnerAllowOnlyOneWinnerPerRound” and “multiWinnerAllowMultipleWinnersPerRound” contests.

   d. note: 0 is valid only when winnerElectionMode is set to "bottomsUpUsingPercentageThreshold"
   e. value: [0..number of declared candidates]
5. "minimumVoteThreshold" optional
   a. if a candidate receives fewer than this number of votes in the first round, they are automatically eliminated
   b. example: 150
   c. value: [0..1000000]
   d. if not supplied: no automatic elimination occurs (equivalent to setting it to 0)
   e. 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.
6. "maxSkippedRanksAllowed" required
   a. 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
   b. example: 1
   c. value: [unlimited, 0..1000000]
7. "maxRankingsAllowed" required
   a. 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
   b. example: 15
   c. values: [max, 1..1000000]
8. "rulesDescription" optional
   a. text description of this rules configuration for organizing your config files -- not used by the tabulator
   b. Example "Maine Rules"
   c. value: string of length [1..1000]
9. "batchElimination" optional

a. Tabulator will use batch elimination (only valid for single-winner contests)
b. Value: true | false
c. If not supplied: false

10. "continueUntilTwoCandidatesRemain" optional
    a. tabulator will keep tabulating (beyond winning round) until only two candidates remain (only valid for single-winner contests)
    b. Value: true | false
    c. If not supplied: false

Out of Scope Settings

11. "overvoteDelimiter" optional and can be used only if provider is ES&S; must be blank when overvoteLabel is provided
    a. string that will be used to split a cell into multiple candidate strings in the case of an overvote
    b. example: //
    c. 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
12. "tabulateByPrecinct" optional
    a. Tabulator will generate a results spreadsheet for each precinct
    b. Value: true | false
    c. If not supplied: false
13. "generateCdfJson" optional
    a. Tabulator will generate a JSON of cast vote records in the Common Data Format
    b. Value: true | false
    c. If not supplied: false
    d. "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
14. "nonIntegerWinningThreshold" optional

    a. the vote threshold used to determine winners can be a non-integer
    b. if true, threshold = V/(S+1) + 10^-d
    c. if false, threshold = floor(V/(S+1)) + 1

d. where V = total number of votes (in the current round or in the first round, depending on the winnerElectionMode); S = numberOfWinners; and d = decimalPlacesForVoteArithmetic
e. (note that S+1 in the formulas above becomes just S if hareQuota is set to true.)
f. Only valid for “multiWinnerAllowOnlyOneWinnerPerRound” and “multiWinnerAllowMultipleWinnersPerRound” contests
g. value: true | false
h. if not supplied: false

15. *Disclaimer

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

16. "hareQuota*" optional
    a. the winning threshold should be computed using the Hare quota* (votes divided by seats) instead of the preferred Droop quota (votes divided by (seats+1))
    b. Only valid for “multiWinnerAllowOnlyOneWinnerPerRound” and “multiWinnerAllowMultipleWinnersPerRound” contests
    c. Value: true | false
    d. If not supplied: false
17. "randomSeed" required if tiebreakMode is "random", "previousRoundCountsThenRandom", or "generatePermutation"
    a. the integer seed for the application's pseudorandom number generator
    b. value: [-140737488355328..140737488355327]
18. "multiSeatBottomsUpPercentageThreshold" required if winnerElectionMode is "bottomsUpUsingPercentageThreshold" and numberOfWinners is 0
    a. the percentage threshold used to determine when to stop the tabulation and declare winners
    b. note: only valid when winnerElectionMode is "bottomsUpUsingPercentageThreshold" and numberOfWinners is 0
    c. value: [1..100]
19. "decimalPlacesForVoteArithmetic" required
    a. number of rounding decimal places when computing winning thresholds and fractional vote transfers

b. note: only editable when winnerElectionMode is "multiWinnerAllowOnlyOneWinnerPerRound" or "multiWinnerAllowMultipleWinnersPerRound"
c. value: [1..20]

20. "winnerElectionMode" required
    a. which process the program should apply for selecting the winner(s)
    b. value: "singleWinnerMajority" | "multiWinnerAllowOnlyOneWinnerPerRound" | "multiWinnerAllowMultipleWinnersPerRound" | "bottomsUp" | "bottomsUpUsingPercentageThreshold" | "multiPassIrv"
    c. "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
    d. "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
    e. "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
    f. "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

g. "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)

21. overvoteRule" required
    a. how the program should handle an overvote when it encounters one
    b. value: "alwaysSkipToNextRank" | "exhaustImmediately" | "exhaustIfMultipleContinuing"
    c. "alwaysSkipToNextRank": when we encounter an overvote, ignore this rank and look at the next rank in the cast vote record
    d. "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)
        i. Note: This setting has a bug. Within an overvote contest when one candidate is eliminated and the remaining two or more candidates are eligible within the contest, URCVT counts a vote towards one of these candidates instead of immediately exhausting the CVR record.

Document Revision History

Date	Version	Description	Author
4/21/21	1.0.0	Configuration File Parameters	Chris Hughes