CRT Draft 20050914

Section 1 Preface

This document contains "zeroth draft" text from Core Requirements and Testing (CRT) for the following parts of the next iteration of the VVSG:

- Introductory text, including introduction to new standards architecture (Section 2.1.1)
- Requirements on casting (Section 4.4.2) counting and reporting (Section 4.4.3)
- General software integrity (not security -- Section 4.3.1.1.1) and workmanship requirements (Section 4.3.4)
- Process model (Section 4.5.1)
- Logic model (Section 4.5.2)
- Logic verification (Section 6.3.1)
- Beginnings of test protocols (not including security or usability testing -- Section 6.4)
- Miscellaneous minor sections assigned to CRT
- CRT contributions to sections assigned to other subcommittees that have not yet been transferred
- Rationale for changes (Section 2.1)

Text for hardware and software performance requirements and hardware workmanship requirements is distributed in a separate document.

"Zeroth draft" text is neither complete nor reviewed, and informative text is missing. The goal of this draft is just to show the directions that CRT is taking and provide the opportunity for early feedback.

Notes on work that is yet to be done and problems that need to be fixed are shown highlighted like this. These notes are for our own use and will not appear in the final draft. Similarly, the Impact field of requirements is for our own use and will not appear in the final draft.

Notes applying to the entirety of the document follow.

General: go through [1] and [3], ensure that all useful requirements and informative text are retained somewhere.

Global change: replace each occurrence of "qualification" with "certification" or "testing," as appropriate.

General: there are some non-leaf requirements showing up with test references. Harmonize with Figure 1 by changing one or the other.

Section 2 Guidelines overview

Section 2.1 Description and rationale of significant changes vs. [1] or [3]

Section 2.1.1 Document structure

The VVSG have been restructured to reduce redundancy in the requirements and to reduce fragmentation of requirements.

Figure 1 shows a conceptual model of requirements, including their relationships to activities in the voting process. To structure the VVSG entirely according to the voting process would make it more usable by some readers. Unfortunately, as shown in Figure 1, a given requirement may relate to many activities. To structure the VVSG entirely according to the voting process would create more redundancy rather than less.

The new structure compromises by handling the two most common cases in two separate sections. Those requirements that relate to most or all activities are included in Section 4.3, General Requirements, and are organized by subject area. Those requirements that relate to single activities are included in Section 4.4, Requirements by Voting Activity, and are organized by activity. The case that is not handled elegantly is where a given requirement relates to more than one activity, but less than most activities. These requirements are, unavoidably, duplicated. Before final draft, check whether or not this even occurred and change text if it didn't, or if they were put in the general section instead.

A detailed model of the voting process with many activities is included in Section 4.5.1. For the purpose of structuring the document, the voting process has been simplified to just a few activities: preparation, casting, counting and reporting, and auditing and verification.

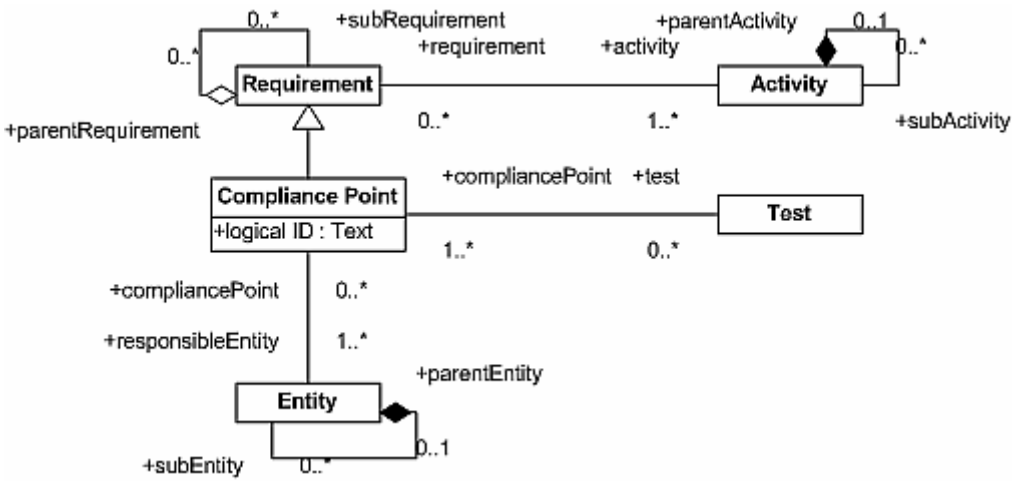

Figure 1 Conceptual model of requirements

Narration of Figure 1

This is a Unified Modeling Language (UML) class diagram with semantics as defined in [4].

Classes: Requirement, Activity, Compliance Point, Test, Entity.

Compliance Point is a subclass of Requirement. Compliance Points have an attribute logical ID of type Text. (A Compliance Point is an identified, testable Requirement.)

Zero to many Requirements in the role subRequirement relate to zero to many Requirements in the role parentRequirement through aggregation. (A requirement may have multiple parents.)

Zero to many Requirements in the role requirement relate to one to many Activities in the role activity.

Zero to many Activities in the role subActivity relate to zero or one Activities in the role parentActivity through composition.

One to many Compliance Points in the role compliancePoint relate to zero to many Tests in the role test.

Zero to many Compliance Points in the role compliancePoint relate to one to many Entities in the role responsibleEntity.

Zero to many Entities in the role subEntity relate to zero or one Entities in the role parentEntity through composition.

Section 2.1.2 Precision and testability

For qualification of voting systems to be consistent, fair, and meaningful, it is necessary to control variability in the conformity assessment system. Testing cannot be an afterthought to a standard: both the requirements to be tested and the methods by which they are to be tested must be specified with appropriate precision. The following hypothetical example illustrates the codependence of requirements and test methods.

Example text	Impact on testing
The unit shall respond to all user input in a timely fashion.	Vague requirement leaves tester in the position of determining what is considered "timely," creates opportunities for inconsistent evaluation and challenges by vendors.

The unit shall respond to all user input in 3 seconds or less.	Good requirement leading to pass-fail verdict. However, the test method to verify the requirement is undefined. Different testing authorities using different test methods may get different results. The vendor could challenge that the set of user inputs chosen by a VSTL is atypical of use in practice.
The VSTL shall measure and report the mean response time and worst response time over the following set of user inputs, employing the test ballot format defined in Section XYZ: opening the ballot; voting for one candidate in each contest; [...]. Units with worst response time exceeding 3 seconds shall be disqualified.	In conjunction with the good requirement, this specified test method enables consistent, informative, and difficult-to-challenge results.

In Resolution #25-05, the TGDC requested that NIST perform a complete review and revision of requirements in the Voting Systems Standards to ensure that they are sufficiently precise to enable meaningful testing and to expand the testing standards to specify test methods for those requirements.

Section 2.1.3 Coding conventions

Volume 1, Section 4.2 and Volume 2, Section 5.4 of the 2002 Voting Systems Standards define coding conventions and a source code review to be conducted by ITAs. Vendors are permitted to use current best practices in lieu of the coding conventions defined in the VSS; however, the coding conventions in the VSS are not aligned with the state of the practice, and if followed, could do more harm than good.

The misalignments are (1) that the conventions, some of which were carried over from the 1990 VSS, are out of date, and (2) that the conventions, limited by their language-independence, are variously incomplete and/or inappropriate in the context of different programming languages with their different idioms and practices.

While they address integrity and maintainability to an extent, the coding conventions are primarily a means to the end of facilitating ITA evaluation of the code's correctness to a level of assurance beyond that provided by black-box testing. That evaluation is underspecified in [1], yielding a cart-before-horse situation in which adherence to the coding conventions could be verified much more rigorously than the correctness of the software.

As part of Resolution #29-05, the TGDC requested that NIST evaluate the [1] software coding standards with respect to their applicability to the recommended standards, and either revise them, delete them, or recommend new software coding standards, as appropriate.

In response, NIST has made recommendations as follows. Coding conventions addressing the need for integrity in voting software have been retained, expanded, and made mandatory, while

stylistic conventions that are made redundant by more recent, publicly available coding conventions have been removed in favor of the published conventions. The requirement that these conventions address maintainability is made clear. Whether the coding conventions addressing integrity can also be replaced by recent, publicly available coding conventions for high-integrity software is yet to be determined.

One possibly controversial recommendation included in the changes is to require the use of a programming language that supports structured exception handling. This rules out the C language, which remains in wide use, and forces a migration to a descendant language, namely C++, C#[1] or Java. Similarly, older versions of Visual Basic that lacked structured exception handling are superseded by Visual Basic .NET.

This recommendation is induced by existing requirements in the VSS, specifically:

> I.2.2.5.2.2.g. Nested error conditions shall be corrected in a controlled sequence such that system status shall be restored to the initial state existing before the first error occurred.
>
> I.4.2.3.e. Each module shall have a single entry point, and a single exit point, for normal process flow. ... The exception for the exit point is where a problem is so severe that execution cannot be resumed. In this case, the design must explicitly protect all recorded votes and audit log information and must implement formal exception handlers provided by the language.

It appears to be the intent of these requirements that the voting system software should (A) exhibit behaviors that are representative of structured exception handling, and (B) accomplish these using "formal exception handlers provided by the language." In context, this is puzzling, since the VSS specifically allowed languages that did not support any semblance of formal exceptions. However, as of 2005, programming languages supporting structured exceptions are widely available and widely used, and they contain other refinements and evolutionary advances, relative to their exceptionless ancestors, that contribute to enhanced software integrity, maintainability, and understandability. To require the use of structured exceptions now is in the same spirit of best practices as the VSS' 1990 requirement for structured control constructs. The alternative is to accept less readable source code and a higher likelihood of masked failures.

Though potentially painful, the migration from languages not supporting structured exceptions is facilitated by closely related languages that evolved from one another: C and C++, C# or Java, Visual Basic and Visual Basic .NET.

Section 2.1.4 Logic verification

Traditionally, testing methods have been divided into black-box and white-box test design. Neither method has universal applicability; they are useful in the testing of different items.

Black-box testing is usually described as focusing on testing functional requirements, these requirements being defined in an explicit specification. It treats the item being tested as a "black

box," with no examination being made of the internal structure or workings of the item. Rather, the nature of black-box testing is to develop and utilize detailed scenarios, or test cases. These test cases include specific sets of input to be applied to the item being tested. The output produced by the given input is then compared to a previously defined set of expected results.

White-box testing (sometimes called clear-box testing to suggest a more accurate metaphor) allows one to peek inside the "box," and focuses specifically on using knowledge of the internals of the item being tested to guide the testing procedure and the selection of test data. White-box testing can discover extra non-specified functions that black-box testing wouldn't know to look for and can exercise data paths that would not have been exercised by a fixed test suite. Such extras can only be discovered by inspecting the internals.

Complimentary to any kind of testing is logic verification, in which formal methods are used to prove that the logic of the system satisfies certain assertions. When it is impractical to test every case in which a failure might occur, formal methods can be used to prove the correctness of the logic generally. However, verification is not a substitute for testing because there can be faults in a formal proof just as surely as there can be faults in a system. Used together, testing and verification can provide a high level of assurance that a system's logic is correct.

This revision of the Voluntary Voting System Standards adds logic verification to the testing campaign to achieve a higher level of assurance.

Section 2.1.5 Public Information Package (PIP)

Public assurance that the voting system is fit for use can occur vicariously, through trust in the VSTL and election officials; indirectly, through verification that the qualification process was responsibly executed; directly, through election verification; or through a combination of these.

In Resolution #28-05, the TGDC requested that NIST recommend standards on data to be provided, called a "Public Information Package," that must be publicly available and published as evidence that the qualification process was responsibly executed. These requirements now appear in Section 5.4.

Section 2.2 List of sources reviewed

Revise to cite all of the state laws and whatever else that have been reviewed.
Should we mention the test reports? (Do not contain requirements *per se*)

Section 2.2.1 Review of existing standards, specifications, and related work

To ensure that previously written requirements would not be overlooked, NIST reviewed the following resources. The resulting guide to existing requirements has not been put into publishable form but is being utilized by project members as they develop new recommendations.

NIST also reviewed sample ballot formats, vote data reports and other materials from several states.

Section 2.2.1.1 Standards, draft standards, regulations, and guidelines

[HAVA] Help America Vote Act of 2002, Public Law 107-252, 2002-10-29.

[2002VSS] 2002 Voting Systems Standards, available from http://www.fec.gov/pages/vssfinal/vss.html.

[P1583/D5.3.1] IEEE Draft Standard for the Evaluation of Voting Equipment, draft 5.3.1, 2004-10-08, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[CoE 2004-09-30] Council of Europe, Committee of Ministers to member states on legal, operational, and technical standards for e-voting, adopted by the Committee of Ministers on 2004-09-30 at the 898th meeting of the Minister's Deputies, e-mail from Lori Steele, 2004-11-10.

[EML3] Election Markup Language v3.0, 2003-02-24, available from http://www.oasis-open.org/committees/election/index.shtml. EML4 is out -- read it.

[SP 500-256] Sharon J. Laskowski et al., "Improving the Usability and Accessibility of Voting Systems and Products," NIST SP 500-256, 2004-05, available from http://www.vote.nist.gov/Final%20Human%20Factors%20Report%20%205-04.pdf.

[508] Section 508 of the Rehabilitation Act: Electronic and Information Technology Accessibility Standards, 2000-12-21, available from http://www.access-board.gov/508.htm.

[ADA] ADA Checklist for Polling Places, 2004-02, available from http://www.usdoj.gov/crt/ada/votingchecklist.htm.

Section 2.2.1.2 Issue lists

Update to 5.3.2b?

[D5.3.1 Comments 2004-10-19] Comments for d5-3-1 dated 10-19-2004 revC.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Software Comments 2004-09-01] Software comments 5.0 (9-01-04).xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Security Comments 2004-08-18] Security extract V5 Comments - 2nd NJ Meeting.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Reliability Accuracy Comments 2004-09-06] 5.0 Comments Section 5.2 & 6.2 (9-6-04).xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Accessibility Comments 2004-08-01] V5 Ballot Accessibility Comments - TG3 (8-1-04) .xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Environmental 2004-08-15] 5.0 Comments Section 5.4 & 6.4.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 EMC 2004-08-23] 5.0 Comments Section 5.5 (8-23-04).xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Provisional 2004-09-10] Gough-Provisional Ballot Comments.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 COTS 2004-06-18] Resolutions for COTS Comments for Draft 5.0 of IEEE P-1583, http://www.lipsio.com/COTS/docs/COTS.resolved.html.

[5.0 TDP 2004-04-23] 5.0 p1583 _TDP-Proposed resolution_Apr04.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

[5.0 Comments 2003-10-16] Ballot Comment Form 5-0 10-16-2003.xls, available from http://grouper.ieee.org/groups/scc38/1583/private/history_041019.html (password-protected).

Section 2.2.1.3 Requests for proposals

[AZ] "OCR and DRE Voting Equipment - Statewide," Request for Proposal, Arizona, 2003. E-mail from Allan Eustis, 2004-10-12.

[CO-REG] "Statewide Voter Registration System," Request for Proposals # DOS-HAVA-0001, Colorado, 2004-01-16, formerly available from http://www.sos.state.co.us/pubs/hava/hava_main.htm (now gone).

[CO-IVV] "Independent Verification and Validation for SCORE Project," Request for Proposals # DOS-HAVA-0002, Colorado, 2004-06-03, formerly available from http://www.sos.state.co.us/pubs/hava/hava_main.htm (now gone).

[GA] Request for Proposal GTA000040, Georgia, 2001. E-mail from Merle King via Allan Eustis, 2004-10-11.

[MD] "Direct Recording Electronic Voting System and Optical Scan Absentee Voting System for Four Counties," Project Number SBE-2002-01, Maryland, 2001-07-17, available from http://www.elections.state.md.us/citizens/voting_systems/voting_system_procurement.html.

[MI] Invitation To Bid # 071I4001011, Michigan, 2003, available from http://www.michigan.gov/sos/0,1607,7-127-1633_11619_27151-77943--,00.html.

[OH-VOT] "Statewide Voting System(s)," Request For Proposal # SOS0428365, Ohio, 2003-05-23, available from http://www.sos.state.oh.us/sos/hava/index.html.

[OH-REG] Request For Proposal # SOS032786279, Ohio, 2003-04-09, available from http://www.sos.state.oh.us/sos/hava/index.html.

[UT] "Executive Summary: Voting Equipment Selection Committee Request for Proposal," Utah. E-mail from Allan Eustis, 2004-10-07.

Section 2.2.1.4 Testimony

[Coney 2004-09-22] Lillie Coney, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Conrad 2004-09-22] Frederick Conrad, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Deutsch 2004-09-21] Herb Deutsch, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Fischer 2004-09-20] Eric A. Fischer, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Gaston 2004-09-20] Charles A. Gaston, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Golden 2004-09-22] Diane Cordry Golden, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Jones 2004-09-20] Douglas W. Jones, testimony to EAC, available from http://www.cs.uiowa.edu/%7Ejones/voting/nist2004.shtml.

[Jones 2004-09-23] Douglas W. Jones, supplemental testimony to EAC, available from http://www.cs.uiowa.edu/%7Ejones/voting/nist2004supp.shtml.

[King 2004-09] Merle S. King, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Noren 2004-09] Wendy S. Noren, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Redish 2004-09-22] Janice Redish, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Relton 2004-09-21] Joy Relton, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Saltman 2004-09-20] Roy G. Saltman, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Shamos 2004-09-20] Michael I. Shamos, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

[Wallach 2004-09-20] Dan S. Wallach, testimony to EAC, available from http://vote.nist.gov/sept04hearings.htm.

Section 3 Terminology standard

Glossary goes here.

archivalness: The length of time that a medium will preserve its content without significant loss.

callable unit: (Of a software program or analogous logical design) A function, method, operation, subroutine, procedure, or analogous structural unit that appears within a module.

cast ballot: Ballot that has been submitted by the voter to election officials for tabulation. See also read ballot and counted ballot.

casting: An act performed by the voter to confirm his or her intent to vote a particular way. Casting is irrevocable and final, regardless of the technology used. For a given ballot, casting and spoiling are mutually exclusive acts.

counted ballot: Ballot that has been processed and whose votes are included in the candidate and measure vote totals. See also cast ballot and read ballot.

error: As used in voting system accuracy testing, an error is defined as the incorrect capturing, recording, storing, consolidation, or reporting of a vote. (Source: Derived from [1] I.3.2.1 and II.C.5.) The word "error" is used in the general sense elsewhere, e.g., in Section 4.3.1.1.2.

failure: As used in voting system reliability testing, a failure is defined as any event which results in (a) loss of one or more functions, (b) degradation of performance such that the device is unable to perform its intended function for longer than 10 seconds, (c) automatic reset, restart or reboot of the voting system, operating system or application software, (d) a requirement for an unanticipated intervention by a person in the role of poll worker or technician before the test can continue, or (e) error messages and/or audit log entries indicating that a failure has occurred. (Source: Expanded from [1] I.3.4.3.)

module: A structural unit of software or analogous logical design, typically containing several callable units that are tightly coupled. Modular design requires that inter-module coupling be

loose and occur over defined interfaces. A module should contain all elements needed to compile or interpret successfully and have limited access to data in other modules. A module should be substitutable with another module whose interfaces match the original module. In software, a module typically corresponds to a single source code file or a source code / header file pair. In object-oriented languages, this typically corresponds to a single class of object.

official data: Define. Taxonomy of data types. See also, unofficial data.

read ballot: Ballot that has been processed but may or may not be counted. For example, an optical scan cast ballot that has been scanned successfully is a read ballot. See also cast ballot and counted ballot.

record: Preserved evidence of activities performed or results achieved (e.g., forms, reports, test results).

report: A self-contained, timestamped, archival document, such as a printout or analogous electronic file, that is produced at a specific time and subsequently protected from modification.[2] A report is a specific kind of record.

unofficial data: Define. Taxonomy of data types. See also, official data.

voting process: The entire array of procedures, people, resources, equipment and locations as associated with the conduct of elections.

voting system: The integrated mechanical, electromechanical, or electronic equipment and software required to program, control, and support the equipment that is used to define ballots; to cast and count votes; to report and/or display election results; and to maintain and produce all audit trail information. It additionally includes the associated documentation used to operate the system, maintain the system, identify system components and their versions, test the system during its development and maintenance, maintain records of system errors and defects, and determine specific changes made after system certification. A voting system may also include the transmission of results over telecommunication networks.

Section 4 Product standard

Section 4.1 Introduction

Section 4.1.1 Structure

NIST is recommending a reorganization of the VVSG to bring them in line with applicable standards practices that are abstracted from our years of association with ISO, W3C and other standards-creating organizations. This includes expanding the *conformance clause* that was added in [3], identifying testable requirements as *compliance points*, and defining *profiles*, which

allow requirements to vary as needed to accommodate variations in voting equipment.

Preferably, requirements should specify *what* (the desired performance), not *how* (a design to accomplish that). For example, a requirement that reads "single-bit errors shall be detected" is preferable to one that reads "products shall use memories with parity bits." Profiles are created to resolve the conflict that occurs when the *what* depends on the *how*. For example, the unstated assumption that the voting equipment would have an electronic memory at all requires placing the preceding example in a profile for electronic voting equipment.

Design-constraining requirements are controversial because vendors would like the freedom to provide the desired qualities / performance in different ways. However, in cases where vendors are unable to determine for themselves whether or not a given design is conforming, they may welcome design constraints as a way to avoid repeated failures and costly retesting of their products. Moreover, in cases where the desired quality is difficult to define abstractly, an enumeration of conforming cases may be the only practical alternative, particularly if there is only one design approach that is ever actually usable in practice. Some pragmatism is required.

A vendor who is submitting a system for testing must make an *implementation statement* that identifies exactly which profiles the system is asserted to support. Conformance tests are catalogued according to which compliance points they exercise. The set of conformance tests appropriate to that system may then be determined automatically. Upon passing those tests, the system may be qualified for only the claimed profiles.

Identified compliance points and a profiles mechanism in the VVSG facilitate traceability from state standards to the VSS. States may define their own profiles over the VVSG, adding compliance points they deem necessary without excessive repetition and revision of VVSG text.

Section 4.2 Conformance Clause

Incorporate and revise from [3] (add profiles, expand implementation statement).

In response to TGDC Resolution #27-05, NIST has eliminated the provision in [1] and [3] for qualification of voting systems that do not conform to the requirements.[3] One member of the TGDC indicated that this provision was of historical origin and is of no further use.

Section 4.2.1 Implementation statement

- Full product identification
- Version of VVSG
- Profiles (see Section 4.2.2)
- Capacities and limits (especially those appearing in Section 4.5.2.1)

Need to add max processing rate for paper-based to capacities and limits.

Section 4.2.2 Profiles

An implementation statement [for a particular device *] shall identify:

- Exactly 1 profile from group 1
- Exactly 1 profile from group 2[4]
- All applicable profiles from group 3

* Currently we have an ambiguity about systems versus devices. The problem case is when you have blended systems, e.g., some precincts are DRE and others are marksense.

```
Profile group 1:  product classes, taxonomy A

Choose 1 of:
  Profile                                       2002VSS reference
  -------                                       -----------------
  Precinct count                                1.5.5
  Central count                                 1.5.6

Profile group 2:  product classes, taxonomy B

Choose at most 1 of:
  Marksense                                     1.5.2
  Punchcard                                     1.5.2
  DRE                                           1.5.3
  Mechanical / lever *

(Derived classes:  Paper-based subsumes Marksense and Punchcard;
Electronic subsumes Marksense and DRE)

Profile group 3:  supported functions
(choose all that apply)

In-person voting                                2.5.2
Absentee voting                                 2.5.2
Provisional / challenged ballots                2.5.2, 2.2.8.2o
Review-required ballots                         2.5.2
Closed primaries                                2.2.8.2a
Open primaries                                  2.2.8.2b
Write-ins                                       2.2.8.2e
Ballot rotation                                 2.2.8.2g
Straight party voting                           2.2.8.2h
  Subclass:
    Cross-party endorsement                     2.2.8.2i
Split precincts                                 2.2.8.2j
N of M voting                                   2.2.8.2k
Cumulative voting                               2.2.8.2m
Ranked order voting                             2.2.8.2n
Remote configuration (incoming data)            5
  Subclass:
    Public network remote configuration
Remote data delivery (outgoing data)            5
  Subclass:
    Public network data delivery                1.5.4, 5
```

```
Unofficial results generation                          2.5.4

Independent Dual Verification (IDV)                    May 9 draft, I.6.0.1
  Subclasses:
    Voter-Verified Paper Audit Trail (VVPAT)           May 9 draft, I.6.0.2
    Split Process                                      May 9 draft, I.D
    Witness                                            May 9 draft, I.D
    End-to-end (cryptographic)                         May 9 draft, I.D
```

* Delete this? Is STS handling these?

Issue: All IDV are electronic voting systems (?). Resolve interactions with group 2.

The class *Remote data delivery* includes all systems in which data are transmitted from individual voting machines to some other machine, regardless of whether or not the target machine is located within the same polling place.

Don't know if we need profiles for these (again, the system vs. device ambiguity gets in the way):

- Registration
- Vote gathering
- Tabulating
- Election management

- Polling place
- Central

Section 4.3 General Requirements

Section 4.3.1 Security and System Integrity

Section 4.3.1.1 System Integrity Management

Section 4.3.1.1.1 Executable code and data integrity[56]

General issue: prescriptions for how to write code versus STS-style "open-ended" expert review. To just say "shall have high integrity" is too vague. [1] has some prescriptions; [5] has more. Current direction is to revise and expand the prescriptions in a fairly conservative manner in the expectation that an expert review will follow.

> Requirement 4.3.1.1.1-1 Self-modifying code is prohibited.
>
> Origin: [1] I.4.2.2.

Impact: The VSS text continues "except under the security provisions outlined in section 6.4.e" but there is no 6.4.e.

Requirement 4.3.1.1.1-2 Remotely loaded code is prohibited.

Origin: [5] Section 5.6.2.2.

Impact: This IEEE-originated tightening of the restrictions in [1] I.4.2.2 makes explicit something that was implied in [1] (many requirements about what must be "resident").

Requirement 4.3.1.1.1-3 Dynamically loaded code other than dynamically linked libraries that are a standard part of the platform is prohibited.

Origin: [5] Section 5.6.2.2.

Impact: This IEEE-originated loosening of the restriction in [1] I.4.2.2 is to avoid outlawing Windows, where there is no alternative to DLLs.

Requirement 4.3.1.1.1-4 If interpreted code is used, it shall only be run under a specific, identified version of an industry standard runtime interpreter.

Origin: [5] Section 5.6.2.2.

Discussion: This prohibition is to ensure that the software tested and approved during the qualification process does not change behavior because of a change to the interpreter.

Impact: This IEEE-originated loosening of the restriction in [1] I.4.2.2 is to allow the use of interpreted Java. [1] mentions Java specifically in I.4.2.1 but then prohibits all interpreted code in I.4.2.2.

Requirement 4.3.1.1.1-5 During an election, all systems containing software or firmware shall prevent replacement or modification of executable code (e.g., by other programs on the system, by people physically replacing the memory or medium containing the code, or by faulty code).

Origin: Rewording/expansion of [1] I.4.2.2.

Discussion: This requirement may be partially satisfied through a combination of read-only memory (ROM), the memory protection implemented by most popular COTS operating systems, error checking as described in Section 4.3.1.1.2, and access and integrity controls.

Requirement 4.3.1.1.1-6 All systems containing software or firmware shall prevent access to or manipulation of vote data or audit records (e.g., by other programs on the system, by people physically replacing the memory or medium containing the data,

or by faulty code) except where this access is necessary to conduct the voting process.

Origin: Rewording/expansion of [1] I.4.2.2.

Discussion: This requirement may be partially satisfied through a combination of the memory protection implemented by most popular COTS operating systems, error checking as described in Section 4.3.1.1.2, and access and integrity controls.

Look for overlaps with STS requirements (e.g., shall not put code on removable modules).

Section 4.3.1.1.2 Error checking[76]

General issue: prescriptions for how to write code versus STS-style "open-ended" expert review. To just say "shall have high integrity" is too vague. [1] has some prescriptions; [5] has more. Current direction is to revise and expand the prescriptions in a fairly conservative manner in the expectation that an expert review will follow.

Requirement 4.3.1.1.2-1 All systems shall check information inputs for accuracy, completeness, and validity.

Origin: [7] [SI-10].

Requirement 4.3.1.1.2-1.1 All systems shall ensure that inaccurate, incomplete, or invalid inputs do not lead to irreversible error.

Origin: [1] I.2.2.5.2.2.f.

Requirement 4.3.1.1.2-2 All systems containing software or firmware that is capable of the following types of errors shall check for these errors at run-time and respond defensively when they occur.

a. Arrays or strings with unenforced bounds (includes buffers used to move data);

b. Pointer variable errors;

c. Dynamic memory allocation and management errors;

d. Stack overflow errors;

e. Run-time exception handling errors; Such as double destruction or ambiguous resolution? Need an example. Contact IEEE for clarification.

f. Variables that are not appropriately handled when out of expected boundaries;

g. Known programming language specific vulnerabilities.

Some of these become "should" below. Refactor this, maybe lose the parent requirement.

Origin: [5] Section 5.6.2.2 expansion of [1] I.4.2.2, modified.

Impact: Removed the one about case statements, which is not necessarily an error.

TO DO: Determine whether existing coding conventions for high-integrity software can subsume the following.

Requirement 4.3.1.1.2-2.1 If the software or firmware uses arrays or any analogous data structures and the programming language does not provide automatic run-time range checking of the indices, the indices shall be ranged-checked on every access.

Origin: Expansion of [1] I.4.2.2.

Discussion: All accesses should occur via dedicated accessors (functions, methods, operations, subroutines, procedures, etc.) that range-check the indices, or an equivalent mechanism. Range checking code should not be duplicated before each access.

Impact: Expansion was to specify what constitutes an acceptable "control."

Requirement 4.3.1.1.2-2.2 For languages having pointers or otherwise providing for specifying absolute memory locations, the system should validate pointers or addresses before they are used.

Origin: Slight revision of [5] 6.6.4.2.e.

Discussion: Improper overwriting should be prevented in general as required by Requirement 4.3.1.1.1-5 and Requirement 4.3.1.1.1-6. Nevertheless, even if read-only memory would prevent the overwrite from succeeding, an attempted overwrite indicates a logic fault that must be corrected. Software design should ensure that the validity of the pointer is determinable, and determined, to a greater extent than merely checking that it is not null. Pointer usage that is fully encapsulated within a standard platform library is essentially COTS and is handled per the COTS requirements Dangling ref: COTS.

Impact: This is "should" not "shall" only because it is very difficult in the general case to validate a pointer. It is easier to design the system in such a way that pointers are not required. Should this be a shall?

Requirement 4.3.1.1.2-2.3 If dynamic memory allocation is performed, the software should be instrumented and/or routinely analyzed with an industry standard tool for detecting memory management errors.

Origin: Added precision.

Impact: This is "should" not "shall" only because such tooling may not be available or applicable in all cases.

Requirement 4.3.1.1.2-2.4 What is the prescription to prevent a stack overflow?

Requirement 4.3.1.1.2-2.5 All scalar or enumerated type parameters whose valid ranges as used in a callable unit (function, method, operation, subroutine, procedure, etc.) do not cover the entire ranges of their declared data types shall be range-checked on entry to the unit.

Origin: Elaboration on Requirement 4.3.1.1.2-2.f, which is an expansion of [1] I.4.2.2.

Discussion: This applies to parameters of numeric types, character types, temporal types, and any other types for which the concept of range is well-defined.[8]

Requirement 4.3.1.1.2-2.6 The detection of any of the errors enumerated in this requirement shall be treated as a complete failure of the callable unit in which the error was detected. An appropriate exception shall be thrown and control shall pass out of the unit forthwith.

Impact: This closes the loophole where a vendor might include the mandatory checks but then ignore the results.

Requirement 4.3.1.1.2-2.7 These error checks shall remain active in qualified production code.

Discussion: These errors are incompatible with voting integrity, so masking them is unacceptable. Vendors should not implement error checks using the C++ assert() macro, which is often disabled, sometimes automatically, when software is compiled in production mode.

Impact: This closes the loophole where a vendor might code in such a way that checks get disabled when the software is compiled or run in production mode (as opposed to testing mode).

Requirement 4.3.1.1.2-2.8 Exceptions resulting from failed error checks shall require intervention by election officials (ref role model when available).

Discussion: These errors are incompatible with voting integrity, so masking them is unacceptable.

Impact: This closes the loophole where a vendor might throw the required exceptions but then mask them in an exception handler.

Requirement 4.3.1.1.2-3 When the system can no longer accept another ballot without the potential of overflowing a vote counter or otherwise compromising the integrity of the counts, it shall emit appropriate warnings and audit events and cease to accept new ballots.

Origin: Clarification of [1] II.5.4.2.g.

Discussion: Assuming that the counter size is large enough such that the value will never be reached is not adequate. Vendors are required to state specific limits, and systems are required to react when those limits are reached. Even if the system could fit in more ballots than the documented limit, it is more important that the behavior of the system agree with the documentation and be predictable.

Impact: This closes the loophole where a vendor might include such controls but leave them in a disabled or inactive state.

Test reference: Test 37, Test 38, Test 39, Test 42

Requirement 4.3.1.1.2-3.1 When a system conforming to the *DRE* profile can no longer accept another ballot without the potential of overflowing a vote counter or otherwise compromising the integrity of the counts, it shall emit appropriate warnings and audit events and cease to enable new ballots.

Origin: Clarification of [1] II.5.4.2.g.

Discussion: A DRE shall not initiate a voting session if there is the possibility that the next ballot could not be properly cast and recorded. If there exists a way of voting the ballot that would exceed one of the limits, then the ballot shall not be enabled.

Test reference: Test 37, Test 38, Test 42

Requirement 4.3.1.1.2-4 Systems conforming to the *Precinct count* profile as well as one of the *Marksense* or *Punchcard* profiles shall include a means of identifying failure of the ballot counting device and corrective action needed.

Origin: [1] I.2.4.1.2.2.c.

Requirement 4.3.1.1.2-5 Systems conforming to the *DRE* profile shall include a means of identifying system failure and any corrective action needed.

Origin: [1] I.2.4.1.3.d.

Section 4.3.1.1.3 Exception handling and recovery

This section deals with exception handling and recovery from system failures in general. Requirements specific to the counting of paper ballots, e.g., jamming and multiple feeds, are found in Section 4.4.3.2.

Requirement 4.3.1.1.3-1 Error conditions shall be corrected in a controlled fashion so that system status may be restored to the initial state existing before the error occurred.

Origin: Generalization from [1] I.2.2.5.2.2.g.

Discussion: "Initial state" refers to the state existing at the start of a logical transaction or operation. Transaction boundaries must be defined in a conscientious fashion to minimize the damage. Language changed to "may" because election officials responding to the error condition might want the opportunity to select a different state (e.g., controlled shutdown with memory dump for later analysis).

Impact: This generalization from [1] I.2.2.5.2.2.g (from the nested case to the non-nested case) clarifies the reason we need exception handling.

Requirement 4.3.1.1.3-1.1 Nested error conditions shall be corrected in a controlled sequence so that system status may be restored to the initial state existing before the first error occurred.

Origin: Slight relaxation of [1] I.2.2.5.2.2.g.

Impact: Relaxation was the "shall" to "may" change mentioned in Requirement 4.3.1.1.3-1 discussion.

Requirement 4.3.1.1.3-2 Exceptions and system recovery shall be handled in a manner that protects the integrity of all recorded votes and audit log information.

Origin: Extracted and reworded from [1] I.4.2.3.e.

Requirement 4.3.1.1.3-3 All systems shall be capable of resuming normal operation following the correction of a failure in any component (e.g., memory, CPU, ballot reader, printer) provided that catastrophic electrical or mechanical damage has not occurred.

Origin: Reworded from [1] I.2.2.3.b and c.

Requirement 4.3.1.1.3-4 When recovering from non-catastrophic failure of a device or from any error or malfunction that is within the operator's ability to correct, the system shall restore the device to the operating condition existing immediately prior

to the error or failure, without loss or corruption of voting data previously stored in the device.

Origin: [1] I.2.2.3.a.

Discussion: If, as discussed in Requirement 4.3.1.1.3-1, the system is left in something other than the last known good state for diagnostic reasons, this requirement clarifies that it must revert to the last known good state before being placed back into service.

Section 4.3.1.2 System auditing and event logging

This section is to be provided by STS. The text here is only notes.

Section 4.3.1.2.1 Entry content requirement

See also [1] I.2.2.5.2.1

Section 4.3.1.3 Hardware security

This section is to be provided by STS. The text here is only notes.

Section 4.3.1.3.1 Memory protection requirements

Overlap with Section 4.3.1.1.1 and Section 4.3.1.1.2.

Section 4.3.2 Accessibility, usability, and privacy, general requirements

This section is to be provided by HFP. The text here is only notes.

[1] I.2.2.5.2.2:

All voting systems shall meet the following requirements for error messages:

a. The system shall generate, store, and report to the user all error messages as they occur;

b. All error messages requiring intervention by an operator or precinct official shall be displayed or printed unambiguously in easily understood language text, or by means of other suitable visual indicators;

c. When the system uses numerical error codes for trained technician maintenance or repair, the text corresponding to the code shall be self-contained, or affixed inside the

unit device. This is intended to reduce inappropriate reactions to error conditions, and to allow for ready and effective problem correction; Why should even trained technicians be expected to deal with "guru meditation numbers?" Just ban them.

d. All error messages for which correction impacts vote recording or vote processing shall be written in a manner that is understandable to an election official who possesses training on system use and operation, but does not possess technical training on system servicing and repair;

e. The message cue for all systems shall clearly state the action to be performed in the event that voter or operator response is required;

[... f and g were already handled in other sections.]

[7] [SI-11] Control: The information system identifies and handles error conditions in an expeditious manner.

Supplemental Guidance: The structure and content of error messages should be carefully considered by the organization. User error messages generated by the information system should provide timely and useful information to users without revealing information that could be exploited by adversaries. System error messages should be revealed only to authorized personnel (e.g., systems administrators, maintenance personnel). Sensitive information (e.g., account numbers, social security numbers, and credit card numbers) should not be listed in error logs or associated administrative messages. The extent to which the information system is able to identify and handle error conditions should be guided by organizational policy and operational requirements.

Section 4.3.3 H/W and S/W performance, general requirements

Text for this section is provided separately. Requirements appearing here are only to provide targets for cross-referencing.

[1] I.3.2.5.1.4.b multiple feeds no more than 1 in 10,000.

Section 4.3.3.1 Reliability and Availability (MTBF, MTTR)

Requirement 4.3.3.1-1 All systems shall achieve at least ninety-nine percent availability during normal operation.

Origin: Extrapolated from [1] I.3.4.5.

Discussion: This general requirement is elaborated by Requirement 4.4.2-2.16, Requirement 4.4.2-2.17, Requirement 4.4.3.1-6, Requirement 4.4.3.1-7, Requirement 4.4.3.2-4, and Requirement 4.4.3.4-3.

Section 4.3.3.2 Accuracy/Error Rates

Requirement 4.3.3.2-1 All systems shall achieve an error rate of no more than one in 10,000,000 ballot positions.

Origin: Extrapolated from [1] I.3.2.1.

Discussion: This general requirement is elaborated by Requirement 4.4.2-2.18, Requirement 4.4.2-3.5, Requirement 4.4.3.2-2, Requirement 4.4.3.2-2.1, Requirement 4.4.3.2-3, Requirement 4.4.3.2-5, Requirement 4.4.3.2-6, and Requirement 4.4.3.4-2.2.

Test reference: Section 6.4.1.2.2

Section 4.3.4 Workmanship

Section 4.3.4.1 Engineering practices

Section 4.3.4.1.1 Coding

Section 4.3.4.1.1.1 Selection of programming languages

Requirement 4.3.4.1.1.1-1 Software and firmware associated with the logical, numeric, and interactive operations of voting shall be produced in a high-level programming language with support for structured exception handling, such as Java, C++, C#, Visual Basic .NET, or Ada.

Origin: Rewrite of [1] I.4.2.1.

Discussion: The requirement for the use of high-level language for logical, numeric, and interactive operations does not preclude the use of assembly language for hardware-related segments, such as device controllers and handler programs. Also, operating system software may be designed in assembly language.

Impact: See discussion in Section 2.1.3 regarding structured exceptions. Added "interactive operations" to the list because obfuscating the user interface is as dangerous as obfuscating the tally code. Added firmware to close loophole in hardware/software testing dichotomy. In [9], ES&S wrote: "The NASED Technical Committee has previously ruled that assembler code is permitted as long as the code meets all other requirements. The draft of the IEEE P1583 VSS also makes allowances for use of assembler code as long as software structure requirements are satisfied. We request that NIST and the TGDC consider allowance of assembler code in new or revised source code standards, as long as all other software structural requirements are met."

Section 4.3.4.1.1.2 Selection of general coding conventions

Requirement 4.3.4.1.1.2-1 Software and firmware associated with the logical, numeric, and interactive operations of voting shall consistently adhere to a published, credible set of coding rules, conventions or standards (herein simply called "coding conventions") intended to enhance the workmanship, security, integrity, testability, and maintainability of applications.

Origin: Rewrite of [1] I.4.2.6.

Discussion: Coding conventions that are excessively specialized will not meet the criteria for intent.

Impact: Added "interactive operations" to the list because obfuscating the user interface is as dangerous as obfuscating the tally code. Added firmware to close loophole in hardware/software testing dichotomy.

Requirement 4.3.4.1.1.2-1.1 Coding conventions shall be considered published if they appear in a publicly available book, magazine, journal, or new media with analogous circulation and availability, or if they are publicly available on the Internet.

Origin: Clarification of [1] I.4.2.6.

Discussion: Following are examples of coding conventions that are freely available on the Internet as of 2005-02-17. These are only examples and are not necessarily the best available for the purpose.

- Java: "Code Conventions for the Java™ Programming Language," Sun Microsystems. http://java.sun.com/docs/codeconv/.
- C++: "Programming in C++, Rules and Recommendations," Mats Henricson and Erik Nyquist. http://www.doc.ic.ac.uk/lab/cplus/c++.rules/. (A revised and expanded version was published in Industrial Strength C++, Prentice-Hall, 1996.)
- C#: "Design Guidelines for Class Library Developers," Microsoft. http://www.msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconnetframeworkdesignguidelines.asp.

Impact: Clarification of "published, reviewed, and industry-accepted" language from [1] I.4.2.6.

Requirement 4.3.4.1.1.2-1.2 Coding conventions shall be considered credible if at least two different organizations with no ties to the creator of the rules or to the vendor seeking qualification independently decided to adopt them and made active use of them at some time within the three

years before qualification was first sought.

Origin: Clarification of [1] I.4.2.6.

Discussion: If the "three year rule" was satisfied at the time that a system was first submitted for qualification, it is considered satisfied for the purpose of subsequent requalifications of that system. However, new systems must meet the three year rule as of the time that they are first submitted for qualification, even if they reuse parts of older systems.

Impact: Clarification of "published, reviewed, and industry-accepted" language from [1] I.4.2.6.

Section 4.3.4.1.1.3 Additional requirements

Section 4.3.4.1.1.3.1 Software modularity and programming

Requirement 4.3.4.1.1.3.1-1 Voting system application software shall be designed in a modular fashion.

Origin: Extracted and revised from [1] I.4.2.3.

Discussion: See module. The modularity rules described here apply to the component submodules of a library.

Impact: Removed unclear, untested requirement on COTS. This should not conflict with any coding conventions. In general, I have attempted to distinguish the class-level concept (module) from the method-level concept (for which I use the new term "callable unit"). [1] did not distinguish these adequately.

Requirement 4.3.4.1.1.3.1-1.1 Each module shall have a specific function that can be tested and verified independently of the remainder of the code.

Origin: Extracted and revised from [1] I.4.2.3.a.

Discussion: In practice, some additional modules (such as library modules) may be needed to compile the module under test, but the modular construction allows the supporting modules to be replaced by special test versions that support test objectives.

Requirement 4.3.4.1.1.3.1-1.2 Modules shall be small, easily identifiable, and constructed to be grouped according to functionality.

Requirement 4.3.4.1.1.3.1-1.2.1 No more than 50% of all callable units (functions, methods, operations, subroutines,

procedures, etc.) should exceed 25 lines of code in length (excluding comments and blank lines), no more than 5% of all callable units should exceed 60 lines in length, and no callable units should exceed 180 lines in length.

Origin: Revision of [1] II.5.4.2.i, as revised by Section 6.6.4.2, Paragraph i of [5].

Discussion: "Lines," in this context, are defined as executable statements or flow control statements with suitable formatting.[96]

Impact: Clarified and updated with "module" replaced by "callable unit" and new exclusion for blank lines. We need length limits to keep the logic verification tractable. The original limits, defined in terms of entire modules rather than callable units, missed the mark.

Requirement 4.3.4.1.1.3.1-1.2.2 Initializations of read-only lookup tables shall be exempt from length limitations.

Origin: Relaxation of [1] requirement.

Impact: Resolve unintended consequence of length limits.

Requirement 4.3.4.1.1.3.1-1.2.3 Read-only lookup tables longer than 25 lines should be placed in separate files from other source code if the programming language permits it.

Section 4.3.4.1.1.3.2 Control constructs

Requirement 4.3.4.1.1.3.2-1 In software and firmware modules associated with the logical, numeric, and interactive operations of voting, unstructured programming is prohibited.

Origin: Generalization and summary of [1] I.4.2.4 and II.5.4.1.

Requirement 4.3.4.1.1.3.2-1.1 Neither GoTo nor any equivalent construct shall be used.

Origin: Generalization and summary of [1] I.4.2.4 and II.5.4.1.

Discussion: "Equivalent constructs" include intentional exceptions, early exits from loops or other program blocks that serve no other purpose, and similar linguistic evasions. Early exits may be used if their avoidance would require excessive conditionalizing or duplication of code and they are not prohibited by the chosen coding conventions. Such exits should

be commented appropriately.

Impact: Relaxed [1] stance on early exits. Certain design patterns are very messy to implement without them.

Requirement 4.3.4.1.1.3.2-1.2 Unstructured exception handling (e.g., On Error GoTo, setjmp/longjmp, or excessive conditionalizing of code to work around exceptions) is prohibited.

Origin: Extension of [1] requirements for structured programming.

Impact: If structured exceptions were in wide use in 1990, this requirement probably would have been in the VSS already.

Section 4.3.4.1.1.3.3 Comments

Requirement 4.3.4.1.1.3.3-1 All software and firmware modules associated with the logical, numeric, and interactive operations of voting 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. Date of creation; and

g. Change log (revision record).

Origin: Revised from [1] I.4.2.7.a.

Discussion: Header comments and other commenting conventions should be specified by the selected coding conventions in a manner consistent with the idiom of the programming language chosen. In the event that published, credible coding conventions fail to specify the content of header comments, this generic guideline should be applied. 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 qualification, and should go back as far as the first baseline or release that is deemed reasonably coherent.

Impact: Added exceptions, revised language, other nits. The discussion on change logs responds to a known controversy regarding how far back change logs must go.

Section 4.3.4.1.2 Quality assurance

Section 4.3.4.1.3 Configuration management

Text for this section is not yet available. The following text from earlier documents is retained only in case portions of it remain applicable.

Quality assurance and configuration management

Volume 1, Sections 7 and 8 and Volume 2, Section 7 of [1] require the vendor to follow certain quality assurance and configuration management practices and require the ITA to conduct several audits and documentation reviews to ensure that they were followed. The quality assurance and configuration management requirements in the VSS are a means to the end of ensuring that the vendor has followed responsible engineering practices in general, and are not necessarily the best or most up-to-date guidelines for that purpose.

In Resolution #30-05, the TGDC requested that NIST review and analyze quality assurance and configuration management standards and recommend changes to the VVSG based on that analysis.

Since the Voting Systems Standards were first issued, it has become possible for vendors to be certified under ISO 9000 and/or appraised under CMMI [10]. It is not clear whether a separate standard for voting system vendors, in lieu of requiring ISO 9000 certification to a scope of operations appropriate to the purpose of developing voting systems, is any longer necessary or desirable. However, at its January 2005 meeting, the TGDC expressed fear over the expense and administrative burden involved in ISO 9000 compliance. NIST has not yet completed the review and analysis of related standards to determine whether a less expensive alternative exists or whether the existing standards could be retained without sacrificing quality.

Section 4.3.5 Archival requirements

Section 4.3.5.1 Archivalness of media

(See archivalness definition in Section 3.)

Requirement 4.3.5.1-1 All systems shall maintain the integrity of voting and audit data, including Cast Vote Records, during an election and for a period of at least 22 months afterward.

Origin: Reworded from [1] I.2.2.11.

Section 4.3.5.2 Period of retention

See Requirement 4.6-2.

Section 4.3.6 Interoperability

STS: need to harmonize with access control requirements.

Overlap with [3] I.D.3.2.1 (data formats for token objects).

Requirement 4.3.6-1 All systems shall maximize interoperability and integratability with other systems and/or components of other systems.

Origin: Generalized from database design requirements in [1] I.2.2.6, TGDC Resolution #23-05, and some state RFP(s).

Requirement 4.3.6-1.1 All systems shall maximize interoperability and integratability with respect to election programming data and report data (the content of vote data reports, audit reports, etc.).

Origin: Generalized from database design requirements in [1] I.2.2.6, TGDC Resolution #23-05, and some state RFP(s).

Requirement 4.3.6-1.2 Systems conforming to the *DRE* profile shall maximize interoperability and integratability with respect to ballot image data.

Origin: Generalized from database design requirements in [1] I.2.2.6, TGDC Resolution #23-05, and some state RFP(s).

Requirement 4.3.6-1.3 The interoperability and integratability requirement may be met by providing the capability to export data in a non-proprietary, open standard format.

Origin: Drill-down from TGDC Resolution #23-05.

Requirement 4.3.6-1.4 The interoperability and integratability requirement may be met by storing data in a documented schema in a COTS or non-proprietary, open source database in such a manner that other applications can read and interpret the data.

Origin: Drill-down from [1] I.2.2.6.

Section 4.4 Requirements by voting activity

Add informative material from [1]

Section 4.4.1 Preparing for election and voting

Section 4.4.1.1 Election programming

Import requirements from [1] 2.3.

The Election Management System (EMS) is used to prepare ballots and programs for use in casting and counting votes, and to consolidate, report, and display election results.

There are significant variations among the election laws of the 50 states with respect to permissible ballot contents, voting options, and the associated ballot counting logic.

> Requirement 4.4.1.1-1 The EMS shall enable election officials or their designees [ROLES] to define political subdivision boundaries and multiple election districts.
>
> Origin: [1] I.2.2.6.a.
>
> Test reference: Test 4
>
> Impact: Database references handled by Section 4.3.6.
>
> Requirement 4.4.1.1-2 The EMS shall enable election officials or their designees [ROLES] to define contests, candidates, and issues using all voting variations indicated in the implementation statement.
>
> Origin: [1] I.2.2.6.b, I.2.2.8.2.
>
> Test reference: Need an "issues" test
>
> Impact: Database references handled by Section 4.3.6.
>
> > Requirement 4.4.1.1-2.1 In all systems, the Election Management System shall allow the definition of 1-of-M contests and general elections.
> >
> > > Requirement 4.4.1.1-2.1.1 In all systems, the Election Management System shall allow the definition of contests where the voter is allowed to choose at most one candidate from a list of candidates.
> > >
> > > Origin: Implicit in [1].

Test reference: Test 2, Test 3, Test 20, Test 23

Requirement 4.4.1.1-2.1.2 In all systems, the Election Management System shall allow the definition of political parties and the indication of the political parties (if any) that endorsed each candidate.

Origin: Implicit in [1].

Test reference: Test 2, Test 3, Test 20, Test 23

... and so on through the voting variations. Unclear whether there is value added in repeating these for EMS. Are there any special EMS concerns other than general support? See casting, counting, reporting.

Section 4.4.1.2 Ballot preparation and production

Import requirements from [1] 2.3.

Section 4.4.1.2.1 EMS functions

Requirement 4.4.1.2.1-1 The EMS shall enable election officials or their designees to define ballot formats and select voting options.

Origin: [1] I.2.2.6.c.

Impact: Database references handled by Section 4.3.6.

Test reference: Test 24 and all other tests.

Requirement 4.4.1.2.1-2 The EMS shall enable election officials or their designees to generate ballots and election-specific programs for vote recording and vote counting equipment.

Origin: [1] I.2.2.6.d.

Impact: Database references handled by Section 4.3.6.

Section 4.4.1.2.2 Any issues on the printing of paper ballots

See Requirement 4.6-1.

Section 4.4.1.3 Equipment preparation

Section 4.4.1.3.1 Software installation

Requirement 4.4.1.3.1-1 The EMS shall enable election officials or their designees to install ballots and election-specific programs.

Origin: [1] I.2.2.6.e.

Impact: Database references handled by Section 4.3.6.

Requirement 4.4.1.3.1-2 The EMS shall enable election officials or their designees to test that ballots and programs have been properly prepared and installed.

Origin: [1] I.2.2.6.f.

Impact: Database references handled by Section 4.3.6.

Section 4.4.1.4 Equipment security and integrity

Section 4.4.1.4.1 *In situ* logic and accuracy testing

Decided at STS telecon 20050907 that L&A testing from [1] is CRT task. Fill this in.

Section 4.4.1.5 Opening polls

Requirement 4.4.1.5-1 Systems conforming to the *Precinct count* profile shall provide an internal test or diagnostic capability to verify that all of the polling place tests specified in Section 4.4.1.4 have been successfully completed.

Origin: [1] I.2.4.1.1.a.

Requirement 4.4.1.5-2 Systems conforming to the *Precinct count* profile shall provide for automatic disabling of any device that has not been tested until it has been tested.

Origin: [1] I.2.4.1.1.b.

Requirement 4.4.1.5-3 Systems conforming to the *Marksense* or *Punchcard* profiles shall include a means of verifying that ballot punching or marking devices are properly prepared and ready to use.

Origin: [1] I.2.4.1.2.1.a.

Requirement 4.4.1.5-4 Systems conforming to the *Precinct count* profile as well as one of the *Marksense* or *Punchcard* profiles shall include a means of activating the ballot counting device.

Origin: [1] I.2.4.1.2.2.a.

Requirement 4.4.1.5-5 Systems conforming to the *Precinct count* profile as well as one of the *Marksense* or *Punchcard* profiles shall include a means of verifying that the ballot counting device has been correctly activated and is functioning properly.

Origin: [1] I.2.4.1.2.2.b.

Requirement 4.4.1.5-6 Systems conforming to the *DRE* profile shall include a security seal, a password, or a data code recognition capability to prevent the inadvertent or unauthorized actuation of the poll-opening function.

Origin: [1] I.2.4.1.3.a.

Requirement 4.4.1.5-7 Systems conforming to the *DRE* profile shall include a means of forcing the execution of poll-opening steps in the proper sequence if more than one step is required.

Origin: [1] I.2.4.1.3.b.

Requirement 4.4.1.5-8 Systems conforming to the *DRE* profile shall include a means of verifying that the system has been correctly activated.

Origin: [1] I.2.4.1.3.c.

Section 4.4.2 Casting

STS: add audit record stuff from [1] I.4.4.3 (in-process audit records).

Requirement 4.4.2-1 Systems conforming to the *DRE* profile shall support activating the ballot.

Origin: [1] I.2.4.

Discussion: The concept of ballot activation, where the machinery may play a part in determining or enforcing limits on who may vote and what they may vote on, presently appears only in DRE systems.

Requirement 4.4.2-1.1 Systems conforming to the *DRE* profile shall enable election officials to control the content of the ballot presented to the voter, whether presented in printed form or electronic display, such that each voter is permitted to record votes only in contests in which that

voter is authorized to vote.

Origin: [1] I.2.4.2.a.

Requirement 4.4.2-1.2 Systems conforming to the *DRE* profile and either the *Closed primaries* or *Open primaries* profiles shall enable the selection of the ballot that is appropriate to the party affiliation declared by the voter in a primary election.

Origin: [1] I.2.4.2.f.

Requirement 4.4.2-1.2.1 In an open primary on a DRE system, the voter should be allowed to choose a party affiliation at the start of the voting session and vote the appropriate ballot format in privacy (i.e., the choice of affiliation should be private as well as the selection of votes on the ballot).

Origin: Clarification or extension of existing requirements.

Discussion: As of 2005, the choice of party is generally made in public view. There is an opportunity here to improve the level of privacy offered to the voter.

Test reference: Test 8

Requirement 4.4.2-1.3 Systems conforming to the *DRE* profile shall activate all portions of the ballot upon which the voter is entitled to vote.

Origin: [1] I.2.4.2.g.

Requirement 4.4.2-1.4 Systems conforming to the *DRE* profile shall disable all portions of the ballot upon which the voter is not entitled to vote.

Origin: [1] I.2.4.2.h.

Requirement 4.4.2-2 All systems shall support the gathering of votes using all voting variations indicated in the implementation statement.

Origin: Extrapolated from [1] I.2.2.8.2 and I.2.4.

Requirement 4.4.2-2.1 All systems shall include a voting booth or similar facility in which the voter may vote in privacy.

Origin: [1] I.2.4.1.2.1.b, generalized to all systems.

HFP: Harmonize with general privacy requirements.

Requirement 4.4.2-2.2 All systems shall record the selection and non-selection of individual vote choices for each contest and ballot measure.

Origin: [1] I.2.4.3.1.c.

> Requirement 4.4.2-2.2.1 Systems conforming to the *DRE* profile shall allow the voter to select his or her preferences on the ballot in any legal number and combination.
>
> Origin: [1] I.2.4.3.3.c.
>
> HFP: Lots of other reqs in [1] I.2.4.3.3 are usability.
>
> Requirement 4.4.2-2.2.2 Systems conforming to the *DRE* profile shall prevent the voter from overvoting.
>
> Origin: [1] I.2.4.3.3.f.
>
> Test reference: Test 40, Test 41
>
> Requirement 4.4.2-2.2.3 Systems conforming to the *Marksense* profile shall allow the voter to mark the ballot to register a vote.
>
> Origin: [1] I.2.4.3.2.1.b.
>
> Requirement 4.4.2-2.2.4 Systems conforming to the *Punchcard* profile shall allow the voter to punch the ballot to register a vote.
>
> Origin: [1] I.2.4.3.2.1.b.

Requirement 4.4.2-2.3 All systems shall support gathering and recording votes in 1-of-M contests and general elections.

Origin: [1] I.2.4. Extended [1] I.2.4.2.e to all systems.

Test reference: Test 2, Test 3, Test 20, Test 23

> Requirement 4.4.2-2.3.1 All systems shall be capable of gathering and recording votes in contests where the voter is allowed to choose at most one candidate from a list of candidates.
>
> Origin: Added precision.

Requirement 4.4.2-2.3.2 All systems shall be capable of indicating the political parties (if any) that endorsed each candidate.

Origin: Added precision.

Probably redundant with forthcoming usability / presentation reqs

Requirement 4.4.2-2.4 Systems conforming to the *Closed primaries* profile shall be capable of gathering and recording votes within a voting process that assigns different ballot formats depending on the registered political party affiliation of the voter and supports both partisan and non-partisan offices.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 7

Requirement 4.4.2-2.5 Systems conforming to the *Open primaries* profile shall be capable of gathering and recording votes within a voting process that assigns different ballot formats depending on the political party chosen by the voter at the time of voting and supports both partisan and non-partisan offices.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 8

Requirement 4.4.2-2.6 Systems conforming to the *Write-ins* profile shall record the voter's selection of candidates whose names do not appear on the ballot and record as many write-in votes as the voter is allowed, per the definition of N(*r*) in Section 4.5.2.

Origin: [1] I.2.4.3.1.d.

Test reference: Test 9, Test 15, Test 29, Test 30, Test 33, Test 34

Impact: Removed untestable reference to state law.

Requirement 4.4.2-2.7 Systems conforming to the *Ballot rotation* profile shall be capable of gathering and recording votes when the ordering of candidates in ballot positions within each contest is variable.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 10

Requirement 4.4.2-2.7.1 DRE systems that enable ballot rotation in a given contest shall alter the ordering of candidates or choices in such a manner that no candidate or choice shall ever have appeared in any particular ballot position two or more times more often than any other.

Origin: Clarification or extension of existing requirements.

Discussion: This is less restrictive than requiring sequential rotation. For a contest of M candidates, the order may be shuffled randomly after each batch of M ballots and rotated sequentially within each batch.

Test reference: Test 10

Requirement 4.4.2-2.8 Systems conforming to the *Straight party voting* profile shall be capable of gathering and recording votes for the slate of candidates endorsed by a given political party as well as individual candidates.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 11, Test 31

Requirement 4.4.2-2.8.1 Systems conforming to the *Cross-party endorsement* profile shall be capable of gathering and recording straight-party votes when a given candidate is endorsed by two or more different political parties.

Origin: Clarification or extension of existing requirements.

Test reference: Test 12

Requirement 4.4.2-2.9 Systems conforming to the *Split precincts* profile shall be capable of gathering and recording votes in a precinct where there are distinct ballot formats for voters from two or more election districts.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 13

Requirement 4.4.2-2.10 Systems conforming to the *N of M voting* profile shall be capable of gathering and recording votes in contests where the voter is allowed to choose up to a specified number of candidates ($N(r) > 1$, per Section 4.5.2) from a list of candidates.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 14, Test 15, Test 22, Test 32, Test 33, Test 34

Requirement 4.4.2-2.11 Systems conforming to the *Cumulative voting* profile shall be capable of gathering and recording votes in contests where the voter is allowed to allocate up to a specified number of votes ($N(r) > 1$, per Section 4.5.2) over a list of candidates however he or she chooses, possibly giving more than one vote to a given candidate.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 16, Test 35

Requirement 4.4.2-2.12 Systems conforming to the *Ranked order voting* profile shall be capable of gathering and recording votes in contests where the voter is allowed to rank candidates in a contest in order of preference, as first choice, second choice, etc.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Test reference: Test 17

Requirement 4.4.2-2.13 Systems conforming to the *Provisional / challenged ballots* profile shall be capable of gathering and recording votes within a voting process that allows the decision whether to count a particular ballot to be deferred until after election day.

Origin: Added precision, based on [1] I.2.2.8.2 and glossary.

Discussion: Unique identification of each provisional/challenged ballot is required. See Requirement 4.4.3.3-1.3.

Test reference: Test 18, Test 36

> Requirement 4.4.2-2.13.1 In systems conforming to the *Provisional / challenged ballots* and *DRE* profiles, the DRE shall provide the capability to categorize each provisional/challenged ballot.
>
> Origin: [5] 5.6.5.2.s.2.[6]
>
> Discussion: Categories (e.g., "regular provisional," "extended hours provisional," "regular extended hours") would be jurisdiction-dependent.
>
> Test reference: NEEDS TEST

Requirement 4.4.2-2.14 Systems conforming to the *Review-required ballots* profile shall be capable of gathering and recording votes within a voting process that requires certain ballots to be flagged or separated for review.

Origin: Extrapolated from [1] I.2.5.2.

Discussion: In some systems and jurisdictions, all ballots containing write-in votes might require flagging or separation for review. Support for the profile indicates that the system can flag or separate ballots in this manner. The reasons for which ballots are flagged or separated are jurisdiction-dependent, but are assumed to be different than provisional/challenged. It is assumed that ballot presentation is unchanged for review-required ballots.

STS and HFP: Flagging/separation offers opportunities for fraud and privacy violation.

Requirement 4.4.2-2.15 Systems conforming to the *DRE* profile shall verify (i.e., actively check and confirm) the correct addition of voter selections to the memory components or persistent storage of the device.

Origin: [1] I.3.2.4.3.3.c, expanded to include persistent storage.

Discussion: "Memory components or persistent storage" includes on-board RAM, flash memory, PROMs, hard disks, optical disks, etc. See also Requirement 4.4.2-2.18 and Requirement 4.4.2-3.5.

Requirement 4.4.2-2.16 Systems conforming to the *Marksense* or *Punchcard* profiles shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) during the recording of voter selections (ballot marking or punching).

Origin: [1] I.3.4.5.a.1.

Discussion: This requirement would not be met, for example, if marking or punching equipment had a tendency to jam frequently, requiring poll worker intervention.

Requirement 4.4.2-2.17 Systems conforming to the *DRE* profile shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) while recording and storing the voter's ballot selections.

Origin: [1] I.3.4.5.b.

Requirement 4.4.2-2.18 For systems conforming to the *DRE* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to

recording the voter selections of candidates and contests into voting data storage.

Origin: [1] I.3.2.1.b.1.

Test reference: Section 6.4.1.2.2

Requirement 4.4.2-3 All systems shall support the casting of a ballot.

Origin: [1] I.2.4. Extended [1] I.2.4.2.e to all systems.

Discussion: This does not entail retaining a ballot image. DREs are required to retain ballot images (see Requirement 4.4.3.5-1.3) but other systems might not.

Requirement 4.4.2-3.1 All systems shall allow each eligible voter to cast a ballot.

Origin: [1] I.2.4.2.b, generalized to all systems. See also Requirement 4.6-3 and Requirement 4.6-4.

Requirement 4.4.2-3.2 Systems conforming to the *Marksense* or *Punchcard* profiles shall include secure receptacles for holding voted ballots.

Origin: [1] I.2.4.1.2.1.c.

Requirement 4.4.2-3.3 Systems conforming to the *Precinct count* and either the *Marksense* or *Punchcard* profile shall allow either the voter or the appropriate election official to place the voted ballot into the ballot counting device.

Origin: [1] I.2.4.3.2.1.c.

Requirement 4.4.2-3.4 Systems conforming to the *Central count* and either the *Marksense* or *Punchcard* profile shall allow either the voter or the appropriate election official to place the voted ballot into a secure receptacle.

Origin: [1] I.2.4.3.2.1.c.

Requirement 4.4.2-3.5 For systems conforming to the *DRE* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to recording voter selections of candidates and contests into ballot image storage independently of voting data storage.

Origin: [1] I.3.2.1.b.2. This relates to the "separate path" design requirement below (Requirement 4.4.2-5 et seq.), to be revised.

Test reference: Section 6.4.1.2.2

Requirement 4.4.2-4 Systems conforming to the *DRE* profile shall prevent modification of the voter's vote after the ballot is cast.

Origin: [1] I.2.4.3.3.n. See also Requirement 4.6-5.

Discussion: See casting.

STS: The following design requirements should be obsoleted. Until then, it is important to retain some requirements for meaningful auditability, and these are the best we have at the moment. Nevertheless, they could stand to be revised.

Requirement 4.4.2-5 Systems conforming to the *DRE* profile shall maintain Cast Vote Records using a process and storage location that differs from the main vote detection, interpretation, processing, and reporting path.

Origin: Reworded from [1] I.2.2.4.2.

Requirement 4.4.2-6 Systems conforming to the *DRE* profile shall provide at least two processes that record the voter's selections that, to the extent possible, are isolated from each other.

Origin: [1] I.3.2.4.3.2.c.1.

Requirement 4.4.2-7 Systems conforming to the *DRE* profile shall record and retain redundant copies of the original ballot image.

Origin: [1] I.2.2.2.2.

end undesirable design reqs

Requirement 4.4.2-8 Systems conforming to the *Precinct count* profile shall prevent the printing of reports and the unauthorized extraction of data prior to the official close of the polling place.

Origin: Reworded from [1] I.2.5.3.2.

Section 4.4.3 Counting and reporting

Section 4.4.3.1 Closing polls

Requirement 4.4.3.1-1 Systems conforming to the *DRE* profile shall prevent access to voted ballots until after the close of polls.

Origin: [1] I.2.4.3.3.r. See also Requirement 4.6-6.

Requirement 4.4.3.1-2 Systems conforming to the *Precinct count* profile shall provide designated functions for closing the polling place.

Origin: Reworded from [1] I.2.5.

> Requirement 4.4.3.1-2.1 Systems conforming to the *Precinct count* profile shall provide a means to prevent the further casting of ballots once the polling place has closed.
>
> Origin: Reworded from [1] I.2.5.1.a.
>
> Requirement 4.4.3.1-2.2 Systems conforming to the *Precinct count* profile shall provide an internal test that verifies that the prescribed closing procedure has been followed and that the device status is normal.
>
> Origin: Reworded from [1] I.2.5.1.b.
>
> Requirement 4.4.3.1-2.3 Systems conforming to the *Precinct count* profile shall include a visible indication of system status (i.e., whether the polls are opened or closed).
>
> Origin: Reworded from [1] I.2.5.1.c.
>
> Requirement 4.4.3.1-2.4 Systems conforming to the *Precinct count* profile shall provide a means to produce a diagnostic test record that verifies the sequence of events and indicates that the extraction of voting data has been activated.
>
> Origin: Reworded from [1] I.2.5.1.d.
>
> Requirement 4.4.3.1-2.5 Systems conforming to the *Precinct count* profile shall provide a means to preclude the unauthorized reopening of the polls once the poll closing has been completed for that election.
>
> Origin: Reworded from [1] I.2.5.1.e.

Requirement 4.4.3.1-3 Systems conforming to the *Precinct count* profile shall provide designated functions for generating post-election reports.

Origin: Reworded from [1] I.2.5.

Requirement 4.4.3.1-4 Systems conforming to the *Precinct count* profile shall consolidate the data contained in each unit into a single report for the polling place when more than one voting machine or precinct tabulator is used.

Origin: Reworded from [1] I.2.5.3.2.

Requirement 4.4.3.1-5 Systems conforming to the *DRE* profile shall, if the consolidation of polling place data is done locally, perform this consolidation in a time not to exceed 5 minutes for each device in the polling place.

Origin: Reworded from [1] I.3.2.6.2.1.

Discussion: For requirements on report content see Section 4.4.3.3.

Requirement 4.4.3.1-6 Systems conforming to the *Precinct count* profile shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) during the consolidation of vote selection data from multiple precinct-based systems to generate jurisdiction-wide vote counts, including storage and reporting of the consolidated vote data.

Origin: [1] 3.4.5.c.

Requirement 4.4.3.1-7 Systems conforming to the *Central count* profile shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) during the consolidation of vote selection data from multiple counting devices to generate jurisdiction-wide vote counts, including storage and reporting of the consolidated vote data.

Origin: [1] 3.4.5.d.

Section 4.4.3.2 Required counting functions

The following requirements apply equally to counting that occurs in the precinct and in the central location.

[1] I.3.2.5.1.2 *et seq.* have "exception handling" in the sense of recovery from paper ballot processing problems like jamming and multiple feeds. Incorporate in this section as appropriate.

Requirement 4.4.3.2-1 All tabulators shall support all voting variations indicated in the implementation statement.

Origin: [1] I.2.2.8.1 plus I.2.2.8.2.

Requirement 4.4.3.2-1.1 All systems shall support tabulating votes in 1-of-M contests and general elections.

Origin: Implicit in [1].

Test reference: Test 2, Test 3, Test 20, Test 23

Requirement 4.4.3.2-1.1.1 All systems shall be capable of tabulating votes, overvotes, and undervotes in contests where the voter is allowed to choose at most one candidate from a list of candidates.

Origin: Implicit in [1].

Requirement 4.4.3.2-1.2 Systems conforming to the *Closed primaries* or *Open primaries* profile shall be capable of tabulating separate totals for each political party.

Origin: Added precision, based on [1] reporting requirements.

Test reference: Test 7, Test 8

Requirement 4.4.3.2-1.3 Systems conforming to the *Write-ins* profile shall be capable of tabulating votes for write-in candidates.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 9, Test 15, Test 29, Test 30, Test 33, Test 34

ISSUE: voting processes with manual / optional processing of ballots with write-in votes. The inclusion of votes from write-in ballots may be outside the system and unverifiable. Can a system within such a process possibly conform to the write-ins profile? See also Section 4.5.2.1.

Requirement 4.4.3.2-1.4 Systems conforming to the *Ballot rotation* profile shall be capable of tabulating votes when the ordering of candidates in ballot positions within each contest is variable.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Discussion: This just means that ballot rotation shall not impact the correctness of the count. A mode of failure would be getting confused about the mapping from ballot positions to candidates.

Test reference: Test 10

Requirement 4.4.3.2-1.5 Systems conforming to the *Straight party voting* profile shall be capable of tabulating straight party votes.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 11, Test 31

Requirement 4.4.3.2-1.5.1 A straight party vote shall be

counted as a vote in favor of all candidates endorsed by the chosen party in each contest in which the voter does not cast an explicit vote.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 11, Test 31

Requirement 4.4.3.2-1.5.2 An explicit vote in a given contest takes precedence over a straight party vote and nullifies the effect of a straight party vote for only that contest.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Discussion: As of 2005, Pennsylvania requires that the straight party vote within a given contest be cancelled automatically when an explicit selection is made (Section 1107-A (3) of Pennsylvania Election Code). Elsewhere it may be permissible to require the voter to unselect the straight party selection in that contest before making a new selection. This detail is a usability issue but is irrelevant to conformance with this requirement.

Test reference: Test 11, Test 31

Requirement 4.4.3.2-1.5.3 Systems conforming to the *Cross-party endorsement* profile shall be capable of tabulating straight-party votes when a given candidate is endorsed by two or more different political parties.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 12

Requirement 4.4.3.2-1.6 Systems conforming to the *Split precincts* profile shall be capable of tabulating votes for two or more election districts within the same precinct.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 13

Requirement 4.4.3.2-1.7 Systems conforming to the *N of M voting* profile shall be capable of tabulating votes, overvotes, and undervotes in

contests where the voter is allowed to choose up to a specified number of candidates (N(r) > 1, per Section 4.5.2) from a list of candidates.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 14, Test 15, Test 22, Test 32, Test 33, Test 34

Requirement 4.4.3.2-1.8 Systems conforming to the *Cumulative voting* profile shall be capable of tabulating votes, overvotes, and undervotes in contests where the voter is allowed to allocate up to a specified number of votes (N(r) > 1, per Section 4.5.2) over a list of candidates however he or she chooses, possibly giving more than one vote to a given candidate.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 16, Test 35

Requirement 4.4.3.2-1.9 Systems conforming to the *Ranked order voting* profile shall be capable of determining the results of a ranked order contest for each round of voting.

Origin: [1] I.2.2.8.1 plus I.2.2.8.2.

Discussion: This requirement is minimal. Since ranked order voting is not currently in wide use, it is not clear what, other than the final result, must be computed.

Test reference: Test 17

Requirement 4.4.3.2-1.10 Systems conforming to the *Provisional / challenged ballots* profile shall be capable of tabulating votes, overvotes, and undervotes in contests where the decision whether to count a particular ballot is deferred until after election day.

Origin: Added precision, based on [1] I.2.2.8.1, I.2.2.8.2 and glossary.

Test reference: Test 18, Test 36

Requirement 4.4.3.2-1.11 Systems conforming to the *Review-required ballots* profile shall be capable of tabulating votes, overvotes, and undervotes from ballots that were flagged or separated for review.

Origin: Extrapolated from [1] I.2.5.2.

Discussion: In some systems and jurisdictions, all ballots containing write-in votes might require flagging or separation for review. Support for the profile indicates that the system can flag or separate ballots in this

manner. The reasons for which ballots are flagged or separated are jurisdiction-dependent, but are assumed to be different than provisional/challenged.

Requirement 4.4.3.2-2 For systems conforming to the *Marksense* or *Punchcard* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to scanning ballot positions on paper ballots to detect selections for individual candidates and contests.

Origin: From [1] I.3.2.1.a.1.

A public comment has been received that recommends adjusting the following two requirements to quantify permissible deviations from the target marking area instead of merely conforming to vendor specifications. This comment is available at http://vote.nist.gov/ecposstatements/AVANTEACCURACY.doc.

Test reference: Section 6.4.1.2.2

Requirement 4.4.3.2-2.1 For systems conforming to the *Marksense* or *Punchcard* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to the detection of punches or marks that conform to vendor specifications.

Origin: [1] I.3.2.5.2.a and I.3.2.6.1.1.

Vendor specifications may not reflect the behavior of actual voters. Quantify the required performance. (Requires research with human subjects)

Test reference: Section 6.4.1.2.2

Requirement 4.4.3.2-2.2 Systems conforming to the *Marksense* or *Punchcard* profile shall ignore, and not record, extraneous perforations, smudges, and folds.

Origin: [1] I.3.2.5.2.b.

Quantify "extraneous" -- how big does an extraneous smudge get before it's considered an intentional mark? (Requires research with human subjects -- need to know if the marks were intentional)

Requirement 4.4.3.2-3 For systems conforming to the *Marksense* or *Punchcard* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to conversion of selections detected on paper ballots into digital data.

Origin: [1] I.3.2.1.a.2 and I.3.2.6.1.1.

Test reference: Section 6.4.1.2.2

Requirement 4.4.3.2-4 Systems conforming to the *Marksense* or *Punchcard* profile shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) while scanning the marks on paper ballots and converting them into digital data.

Origin: [1] I.3.4.5.a.2.

Requirement 4.4.3.2-5 For systems conforming to the *Precinct count* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to consolidation of vote selection data from multiple precinct-based systems to generate jurisdiction-wide vote counts, including storage and reporting of the consolidated vote data.

Origin: Reworded from [1] I.3.2.1.

Test reference: Section 6.4.1.2.2

Requirement 4.4.3.2-6 For systems conforming to the *Central count* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to consolidation of vote selection data from multiple counting devices to generate jurisdiction-wide vote counts, including storage and reporting of the consolidated vote data.

Origin: Reworded from [1] I.3.2.1.

Test reference: Section 6.4.1.2.2

Section 4.4.3.3 Reporting

[2] defined "totally blank ballot" as a special case of undervote that may be separately reported (it is a voting variation, which would become a profile in this document). Since it was removed from [1], implication is that nobody needs it anymore, but it was added back in [5]. Do we need this or not?

Requirement 4.4.3.3-1 All systems shall produce reports that account for all votes on all accepted ballots.

Requirement 4.4.3.3-1.1 Systems shall provide no access path from unofficial electronic reports or files to the storage devices for official data.

Origin: Reworded from [1] I.2.5.4.b.

STS: This requirement belongs to STS, but nobody is quite sure what it means.

Requirement 4.4.3.3-1.2 Vote data reports shall be completely consistent

and error-free, with no discrepancy among reports of voting device data at any level.

Origin: Reworded from [1] I.3.2.6.2.2, extended to all systems.

Test reference: Test 1, Test 25 and all other tests.

Requirement 4.4.3.3-1.2.1 Any discrepancy in reports, regardless of source, shall be resolvable to a procedural error, to the failure of a non-memory device, or to an external cause.

Origin: Reworded from [1] I.3.2.6.2.2.

Discussion: If this requirement is applicable, then the system has failed to satisfy Requirement 4.4.3.3-1.2 and is therefore non-conforming. Nevertheless, in practice it is essential that a specific cause be pinpointed. It is not sufficient merely to eliminate everything other than procedural error, failure of non-memory device, or external cause; it is necessary to identify a specific cause that is one of those.

Requirement 4.4.3.3-1.3 Systems conforming to the *Provisional / challenged ballots* profile shall support the independent acceptance and rejection of individual provisional/challenged ballots.

Discussion: This is meant to rule out the mode of failure in which the IDs assigned to provisional ballots fail to be unique, rendering the system incapable of accepting one without also accepting the others with the same ID.

Test reference: Test 18, Test 36

Requirement 4.4.3.3-1.4 Systems conforming to the *Provisional / challenged ballots* profile shall support the acceptance and rejection of provisional/challenged ballots by category.

Origin: [5] 5.6.5.2.s.3.[6]

Discussion: For "category," see Requirement 4.4.2-2.13.1. The behavior when an individual acceptance/rejection conflicts with a categorical acceptance/rejection is system-dependent and should be documented by the vendor.

Test reference: NEEDS TEST

General statement for Requirement 4.4.3.3-2 through Requirement 4.4.3.3-13

The following compliance points were distilled, refactored, and clarified from overlapping, subtly differing requirements appearing several places in Chapters 2 and 4 of [1], including: I.2.2.2.1.c (produce an accurate report of all votes cast), I.2.2.6.h (printed report of everything in I.2.5), I.2.2.9 (ballot counter), I.2.5.2 (means to consolidate vote data), I.2.5.3.1.a (geographic reporting), I.2.5.3.1.b (printed report of number of ballots counted by each tabulator), I.2.5.3.1.c (contest results, overvotes, and undervotes for each tabulator), I.2.5.3.1.d (consolidated reports including other data sources), I.4.4.4.a (number of ballots cast, using each ballot configuration, by tabulator, precinct, and political subdivision), I.4.4.4.b (candidate and measure totals for each contest, by tabulator), I.4.4.4.c (number of ballots read within each precinct and for additional jurisdictional levels, by configuration, including separate totals for each party in primary elections), I.4.4.4.d (separate accumulation of overvotes and undervotes for each contest, by tabulator, precinct, and additional jurisdictional levels), and I.4.4.4.e (for paper-based systems, the total number of ballots both processed and unprocessable, and the total number of cards read).

Requirement 4.4.3.3-2 Systems conforming to the *Marksense* or *Punchcard* profile shall report the total number of cast ballots at the precinct, election district, and jurisdiction reporting levels, by configuration.

Discussion: In the case of DRE systems, this requirement would be redundant with Requirement 4.4.3.3-3 because every cast ballot is a read ballot (but not necessarily a counted ballot). Only when there is a tangible (paper) ballot is it possible to cast a ballot that is never read. There is no sub-requirement for separate reporting of provisional cast ballots because the system is unlikely to know whether a ballot is provisional until it is successfully read.

Requirement 4.4.3.3-3 All systems shall report the total number of read ballots at each reporting level (tabulator, precinct, election district, and jurisdiction), by configuration.

Requirement 4.4.3.3-3.1 Systems conforming to the *Marksense* or *Punchcard* profile shall, if there are multiple card ballots, report the total number of cards read at the precinct, election district, and jurisdiction reporting levels, by configuration.

Requirement 4.4.3.3-3.2 Systems conforming to the *Closed primaries* or *Open primaries* profiles shall report separate totals for each party in primary elections.

Test reference: Test 7, Test 8

Requirement 4.4.3.3-3.3 Systems conforming to the *Provisional / challenged ballots* profile shall report the total number of provisional read ballots at each reporting level (tabulator, precinct, election district, and jurisdiction), by configuration.

Test reference: Test 18, Test 36

Requirement 4.4.3.3-4 All systems shall report the total number of counted ballots at each reporting level (tabulator, precinct, election district, and jurisdiction), by configuration.

Discussion: See also Requirement 4.4.3.3-5, which breaks down counted ballots by contest.

> Requirement 4.4.3.3-4.1 Systems conforming to the *Closed primaries* or *Open primaries* profiles shall report separate totals for each party in primary elections.
>
> Test reference: Test 7, Test 8
>
> Requirement 4.4.3.3-4.2 Systems conforming to the *Provisional / challenged ballots* profile shall report the total number of provisional counted ballots at each reporting level (tabulator, precinct, election district, and jurisdiction), by configuration.
>
> Test reference: Test 18, Test 36

Requirement 4.4.3.3-5 All systems shall report the number of counted ballots for each N-of-M or cumulative voting contest, at each reporting level (tabulator, precinct, election district, and jurisdiction), per the definition of $K(j,r,t_E)$ in Section 4.5.2.

Discussion: This is by contest, while Requirement 4.4.3.3-4 is the overall count. N-of-M in this context includes the most common type of contest, 1-of-M.

Requirement 4.4.3.3-6 All systems shall report the candidate and measure vote totals for each N-of-M or cumulative voting contest, at each reporting level (tabulator, precinct, election district, and jurisdiction), per the definition of $T(c,j,r,t_E)$ in Section 4.5.2.

Discussion: N-of-M in this context includes the most common type of contest, 1-of-M.

Test reference: Test 25 and all other tests.

Requirement 4.4.3.3-7 All systems shall report the number of overvotes for each N-of-M or cumulative voting contest, at each reporting level (tabulator, precinct, election district, and jurisdiction), per the definition of $O(j,r,t_E)$ in Section 4.5.2.

Discussion: N-of-M in this context includes the most common type of contest, 1-of-M. [1] required the reporting of overvotes even on DRE systems where overvoting is prevented (Requirement 4.4.2-2.2.2); that requirement is retained here, though it may be redundant.

Test reference: Test 6, Test 21, Test 28

> Requirement 4.4.3.3-7.1 All systems shall be capable of producing a consolidated report of the combination of overvotes for any contest that is selected by an authorized official (e.g.; the number of overvotes in a given contest combining candidate A and candidate B, combining candidate A and candidate C, etc.).
>
> Origin: From [1] I.2.2.6.h and I.2.5.3.1.e.
>
> Test reference: Test 6

Requirement 4.4.3.3-8 All systems shall report the number of undervotes for each N-of-M or cumulative voting contest, at each reporting level (tabulator, precinct, election district, and jurisdiction), per the definition of $U(j,r,t_E)$ in Section 4.5.2.

Discussion: N-of-M in this context includes the most common type of contest, 1-of-M.

Test reference: Test 26, Test 27 and other tests with undervotes.

Requirement 4.4.3.3-9 Systems conforming to the *Ranked order voting* profile shall report the candidate and measure vote totals for each ranked order contest for each round of voting/counting at the jurisdiction level.

Discussion: This requirement is minimal. Since ranked order voting is not currently in wide use, it is not clear whether a count must be reported for each permutation of choices, how bogus orderings are reported, or how it would be done at multiple reporting levels.

Test reference: Test 17

Requirement 4.4.3.3-10 Systems conforming to the *In-person voting* profile shall include in-person votes in the consolidated reports.

Discussion: "Include" simply means that the final totals must reflect them. It does not entail separate totals for the different kinds of votes.

Requirement 4.4.3.3-11 Systems conforming to the *Absentee voting* profile shall include absentee votes in the consolidated reports.

Discussion: "Include" simply means that the final totals must reflect them. It does not entail separate totals for the different kinds of votes.

Requirement 4.4.3.3-12 Systems conforming to the *Provisional / challenged ballots* profile shall include votes from accepted provisional / challenged ballots in the consolidated reports.

Discussion: "Include" simply means that the final totals must reflect them. It does not entail separate totals for the different kinds of votes. See also Requirement 4.4.3.3-3.3 and Requirement 4.4.3.3-4.2.

Test reference: Test 18, Test 36

Requirement 4.4.3.3-13 Systems conforming to the *Review-required ballots* profile shall include votes from accepted reviewed ballots in the consolidated reports.

Discussion: "Include" simply means that the final totals must reflect them. It does not entail separate totals for the different kinds of votes.

Requirement 4.4.3.3-14 All systems shall be capable of producing reports of all of the pre-election audit records, system readiness audit records, and in-process audit records defined in Section 4.3.1.2.

Origin: From [1] I.2.2.6.i, I.2.3.6 and I.2.5.3.1.f.

Requirement 4.4.3.3-15 All systems shall provide the capabilities to obtain status and equipment readiness reports from each set of electronic equipment.

Origin: Reworded from [1] I.2.3.4.1.b.

ISSUE: status reports not defined.

Requirement 4.4.3.3-16 Systems conforming to the *Unofficial results generation* profile shall provide only aggregated results in unofficial reports, and not data from individual ballots.

Origin: Reworded from [1] I.2.5.4.a.

Test reference: Test 19

Requirement 4.4.3.3-17 Systems conforming to the *Unofficial results generation* profile shall clearly indicate on each unofficial report or file that the results it contains are unofficial.

Origin: Reworded from [1] I.2.5.4.c.

Test reference: Test 19

Requirement 4.4.3.3-18 All systems shall prevent data from being altered or destroyed by report generation, including data in transportable memory.

Origin: From [1] I.2.2.6.h, I.2.5.3.1.g, and I.2.5.3.2.d.

Section 4.4.3.4 Transmitting results

STS: indicated interest in adding requirements here.

Requirements on *Remote data delivery* may apply even if results are electronically transmitted only within the polling place (see Section 4.2.2).

Requirement 4.4.3.4-1 All systems shall ensure that extracted or duplicated information, including Cast Vote Records extracted from DRE machines, is identical to that on the original storage medium.

Origin: Reworded from Section 5.6.9.2, Paragraph k of [5].[6]

Requirement 4.4.3.4-1.1 All electronic systems shall verify (i.e., actively check and confirm) that information as extracted or duplicated to machine-readable media is identical to that on the original storage medium.

Requirement 4.4.3.4-2 Systems conforming to the *Remote data delivery* profile shall prevent data, including data in transportable memory, from being altered or destroyed by the transmission of results over telecommunications lines, local networks, etc.

Origin: Reworded from [1] I.2.5.3.1 and I2.5.3.2.d.

Requirement 4.4.3.4-2.1 Transmitting results over telecommunications lines, local networks, etc. shall not result in modifications to any vote data in the sending system.

Requirement 4.4.3.4-2.2 For systems conforming to the *Remote data delivery* profile, the acceptable voting system error rate (Requirement 4.3.3.2-1) applies to the transmission of data over telecommunications lines, local networks, etc.

Origin: [1] I.5.2.1.

Discussion: This requirement will typically be met through use of a standard error-correcting protocol that meets Requirement 4.3.3.2-1.

Test reference: Section 6.4.1.2.2

Requirement 4.4.3.4-3 Systems conforming to the *Remote data delivery* profile shall achieve at least ninety-nine percent availability (Requirement 4.3.3.1-1) while transmitting data over telecommunications lines, local networks, etc.

Origin: [1] I.5.2.5.

Section 4.4.3.5 Auditing and verification -- IDV profile

This section is to be provided by STS. The text here is only notes.

Requirement 4.4.3.5-1 All systems shall be auditable by election officials.

Origin: Generalized from many [1] requirements.

Include stuff from [1] I.2.2.5. Make sure got all the audit records.

Requirement 4.4.3.5-1.1 All devices that tabulate ballots shall enable election officials to determine the number of ballots cast so far during a particular test cycle or election at any time during the test cycle or election without disrupting any operations in progress.

Origin: Phrasing the functional requirement that was implied by design requirements in [1] I.2.2.9.

Discussion: [1] I.2.4 refers to separate "election counter" and "life-cycle counter;" the latter was an error (intended to delete).

Requirement 4.4.3.5-1.2 Systems conforming to the *DRE* profile shall maintain an accurate Cast Vote Record of each ballot cast.

Test reference: Test 5

Origin: Reworded from [1] I.2.2.4.2.

Requirement 4.4.3.5-1.3 Systems conforming to the *DRE* profile shall provide a capability to retrieve ballot images in a form that people can read.

Test reference: Test 5

Origin: [1] I.2.2.4.2.b and I.3.2.4.3.2.d.

Impact: Rewrote "readable by humans" in plain English.

Requirement 4.4.3.5-2 All electronic systems shall provide software that monitors the overall quality of data read-write and transfer quality status, checking the number and types of errors that occur in any of the relevant operations on data and how they were corrected.

Origin: [1] I.2.2.2.1.e.

Section 4.5 Reference models

Section 4.5.1 Process model (informative)

Section 4.5.1.1 Introduction

This section contains 16 diagrams describing the elections and voting process. The diagrams are expressed in Unified Modeling Language (UML) version 2.0 [4].

To simplify the diagrams, the following shortcuts have been taken.

- The expansion regions around activities that are performed for every precinct or every voter are not shown.
- When a particular object may or may not exist depending on system and jurisdiction-specific factors (e.g., paper-based vs. DRE), that object is modelled as an optional parameter to an activity. This does not capture the constraint that subsequent activities must wait on this object in those jurisdictions where it applies (i.e., in some jurisdictions it is mandatory).
- Objects that flow downstream in an obvious manner through many activities are not shown as inputs/outputs of all of those activities.
- The propagation of the registration database from one election cycle to the next is not shown. The database appears as an input to the Register voters activity with no indication of its origin.
- Many activities produce reports and other objects that eventually flow into the Archive activity. These flows into the archive are not shown.

Section 4.5.1.2 Diagrams

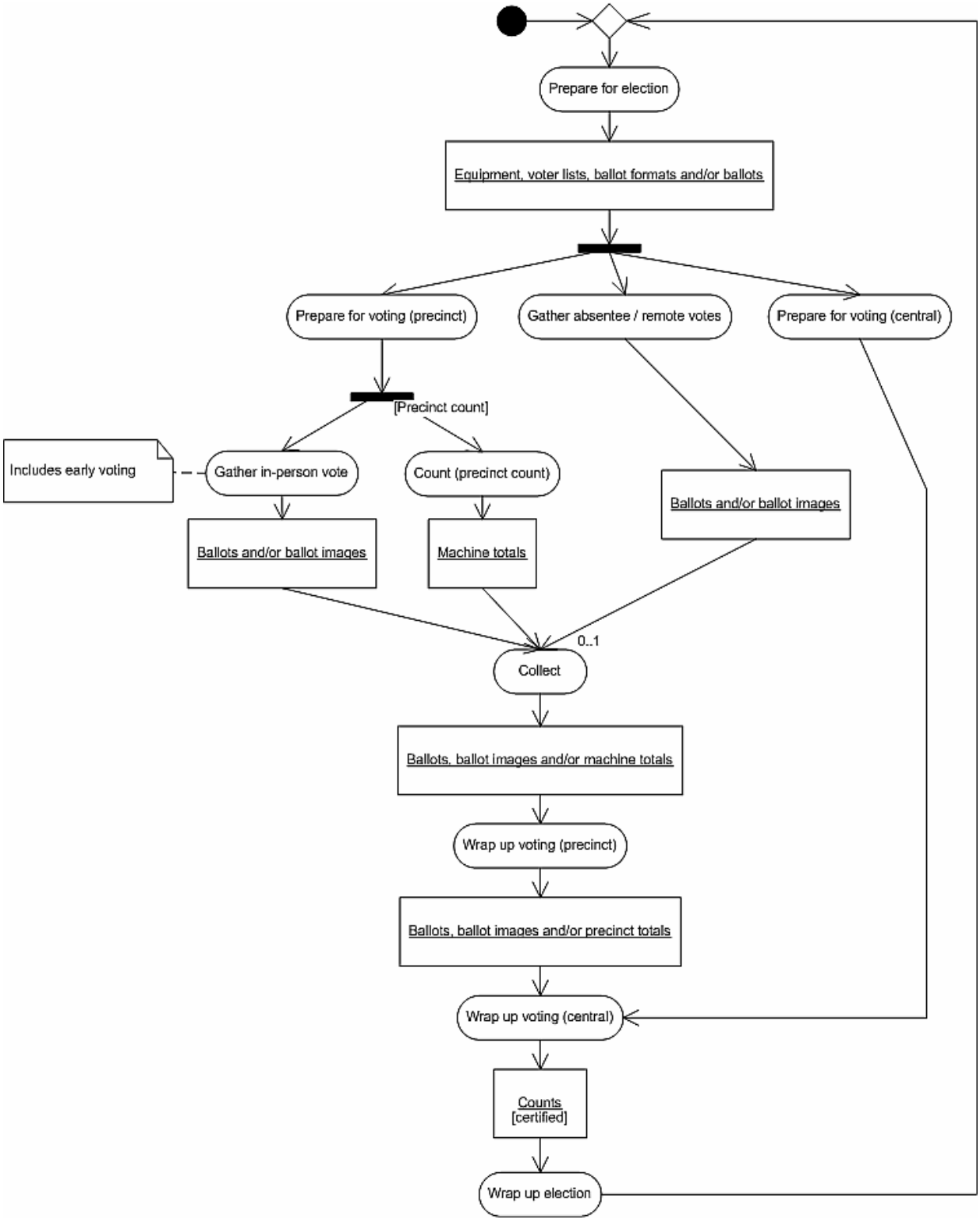

Figure 2 Administer elections

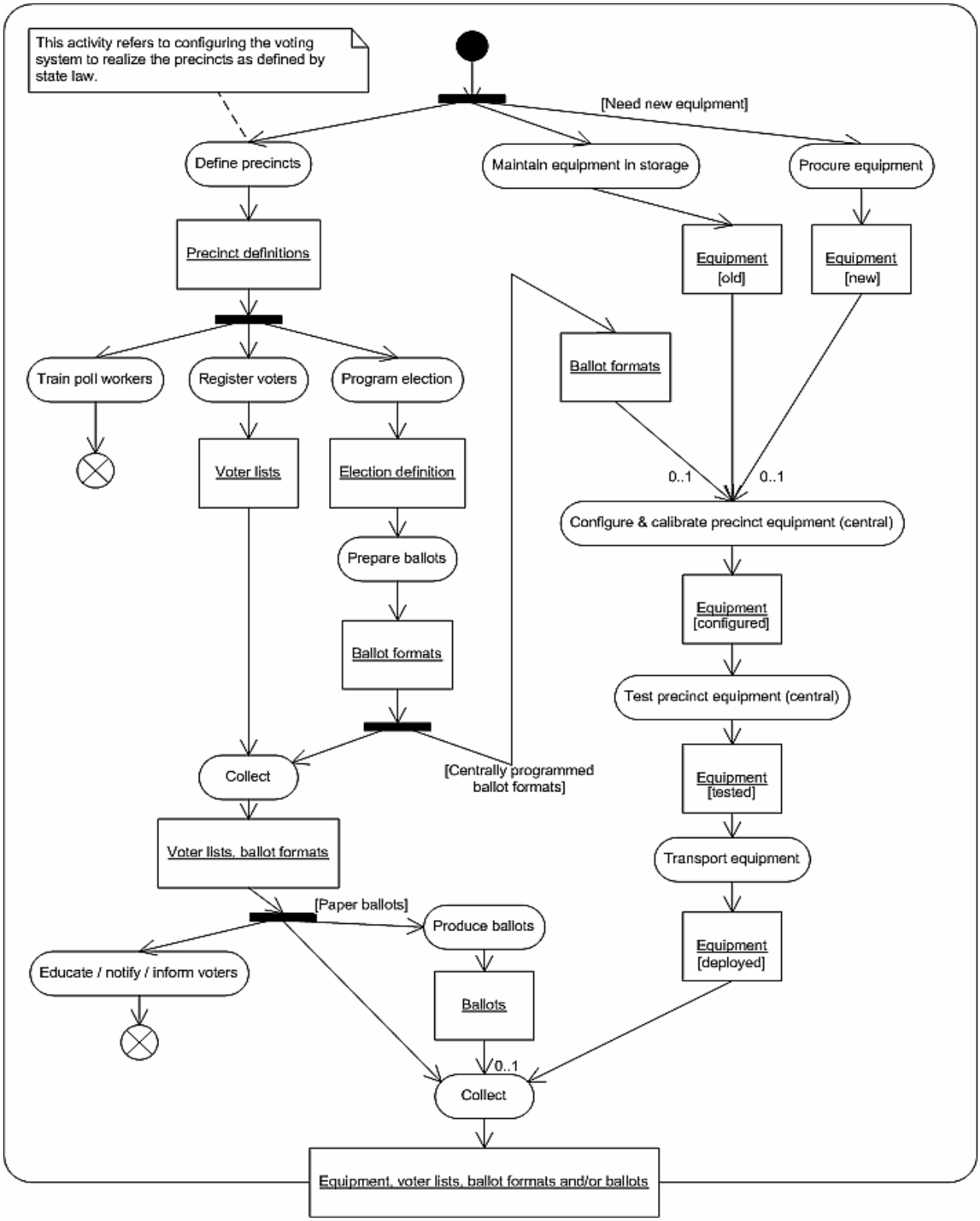

Figure 3 Prepare for election

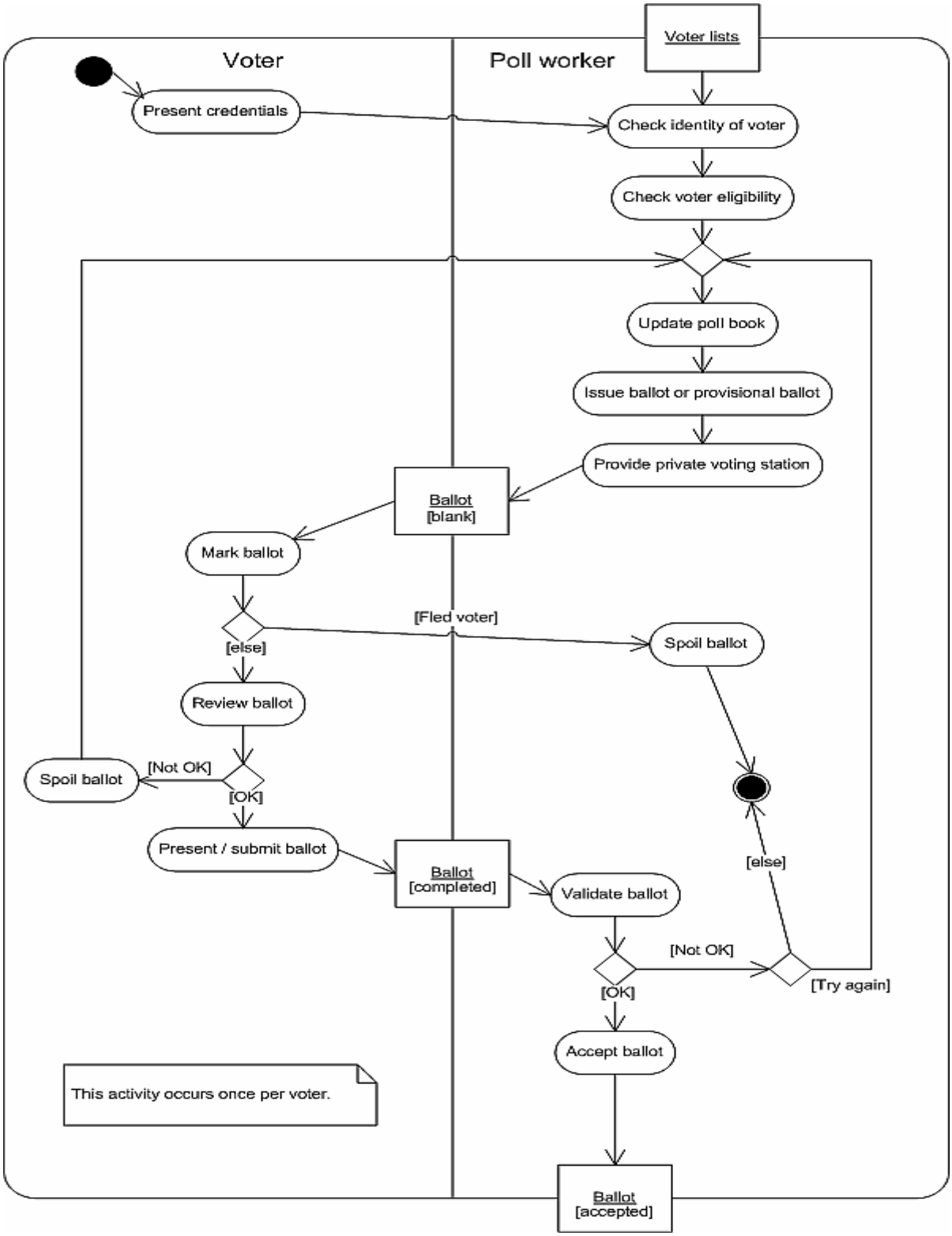

Figure 4 Gather in-person vote (paper-based)

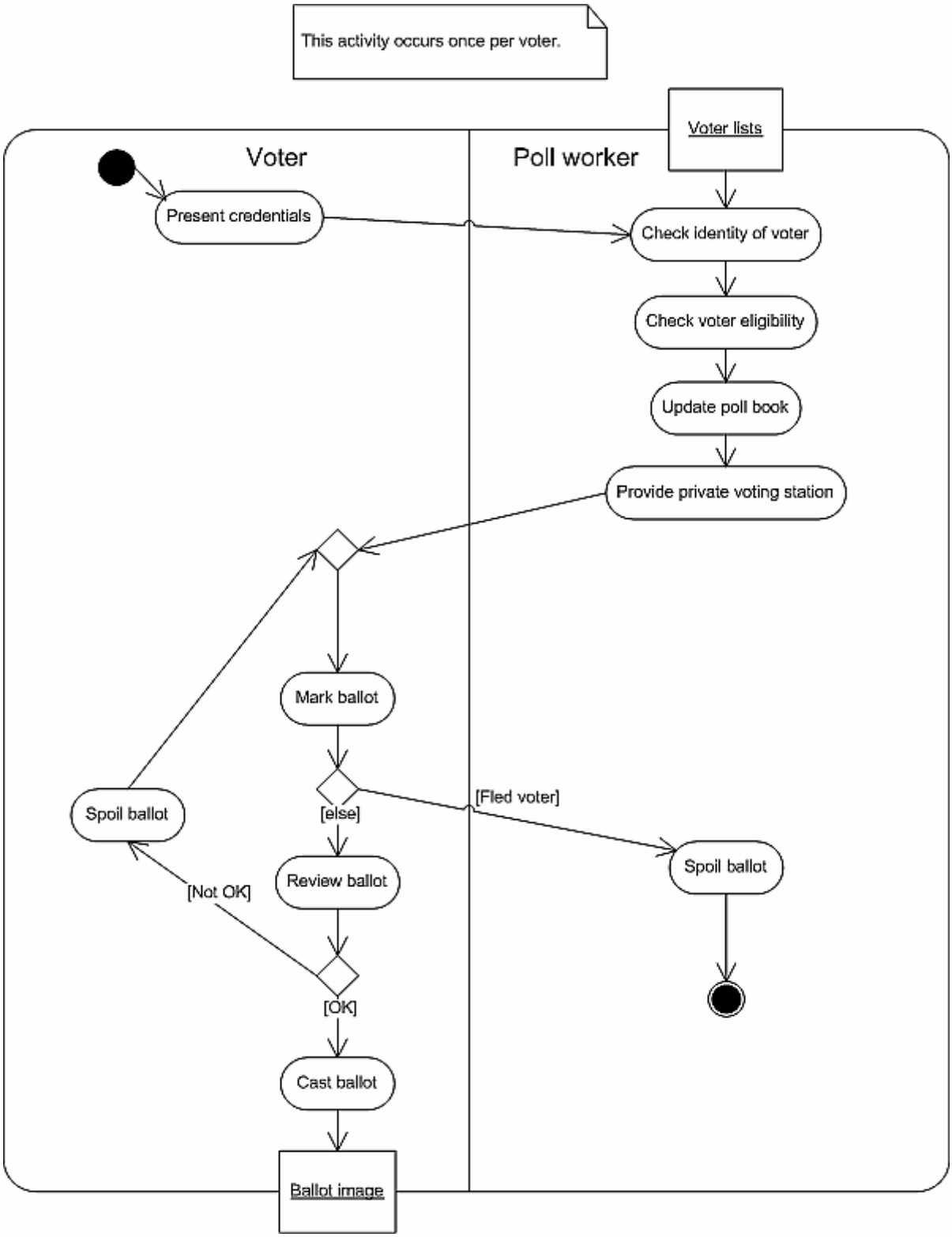

Figure 5 Gather in-person vote (DRE)

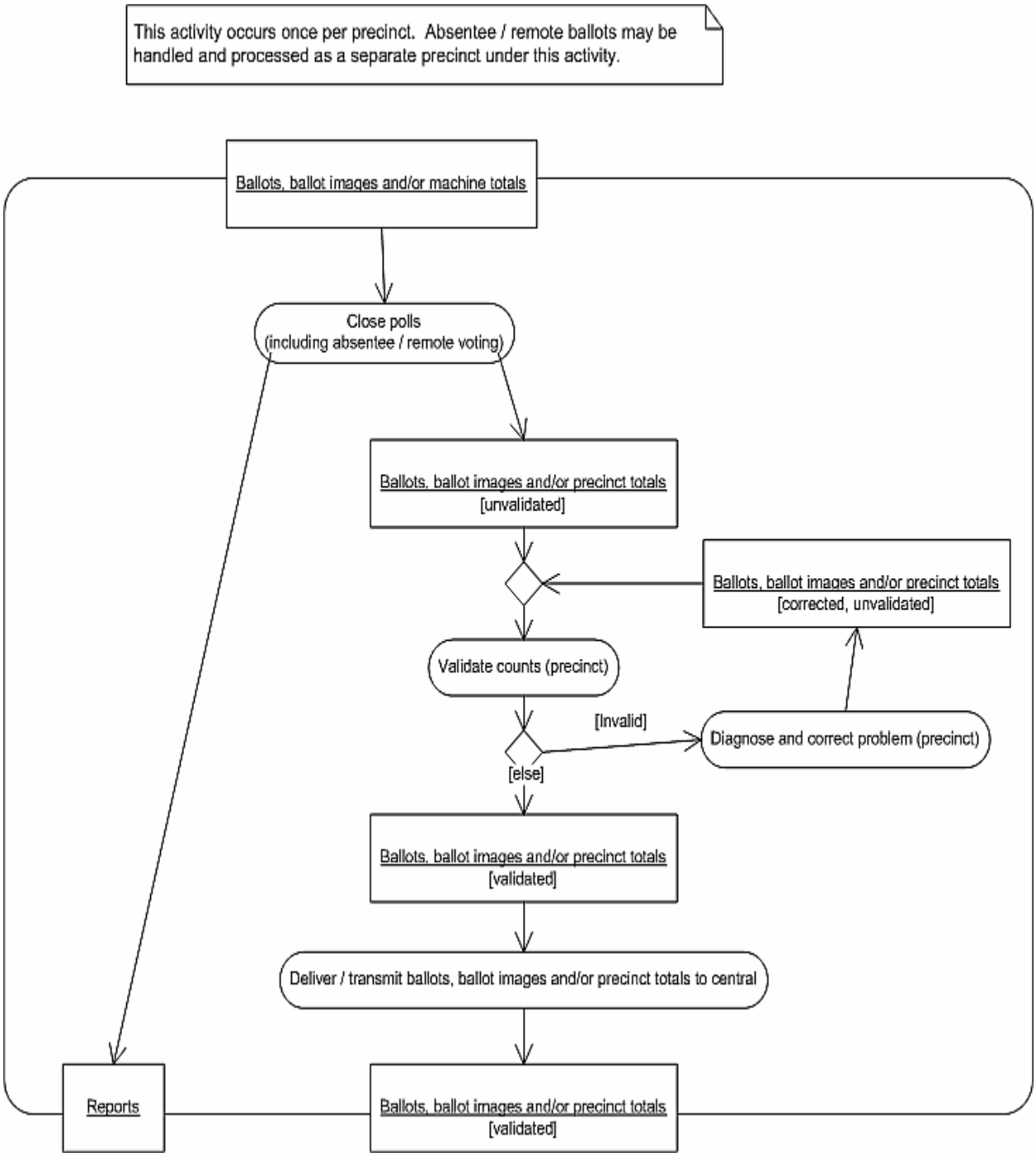

Figure 6 Wrap up voting (precinct)

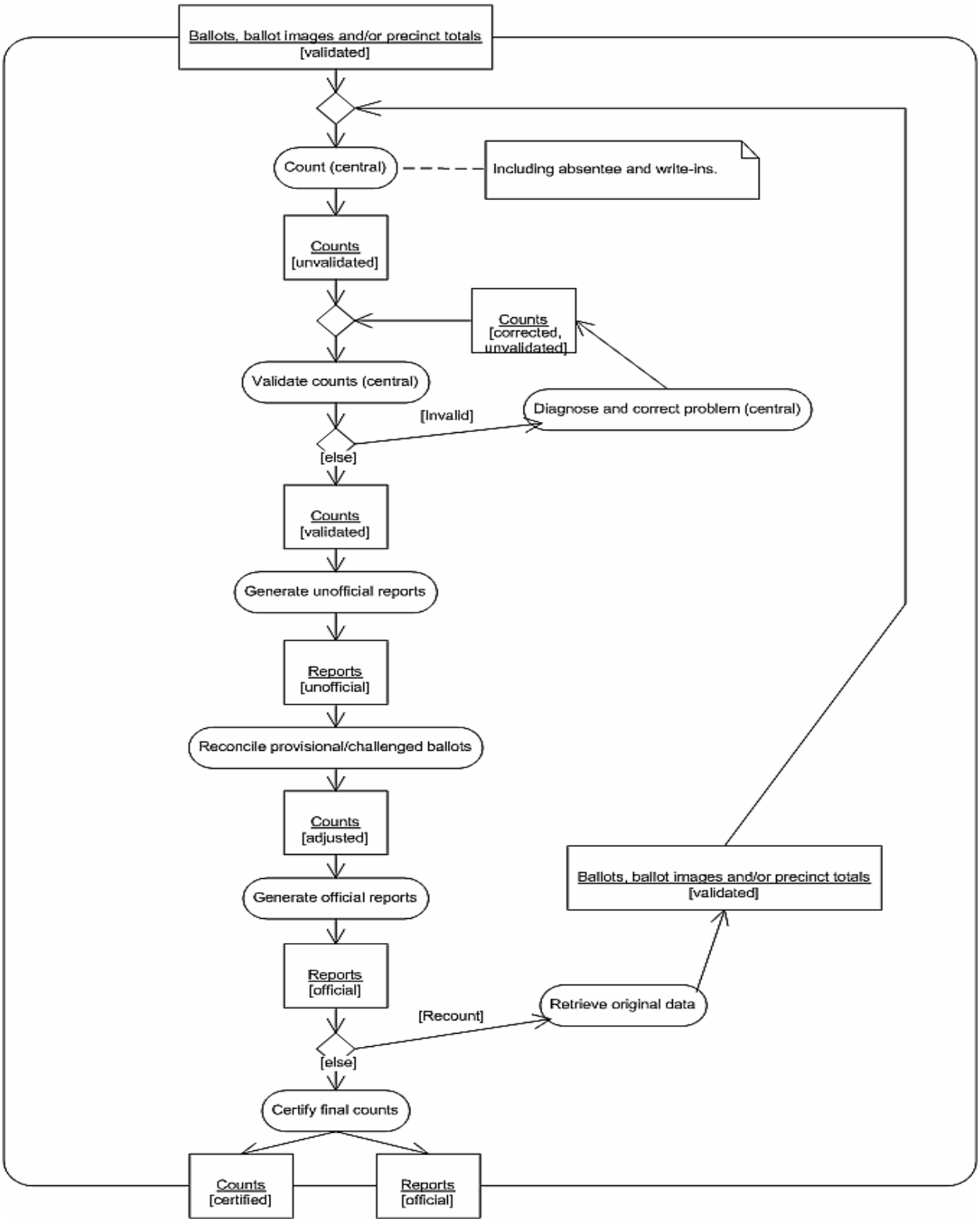

Figure 7 Wrap up voting (central)

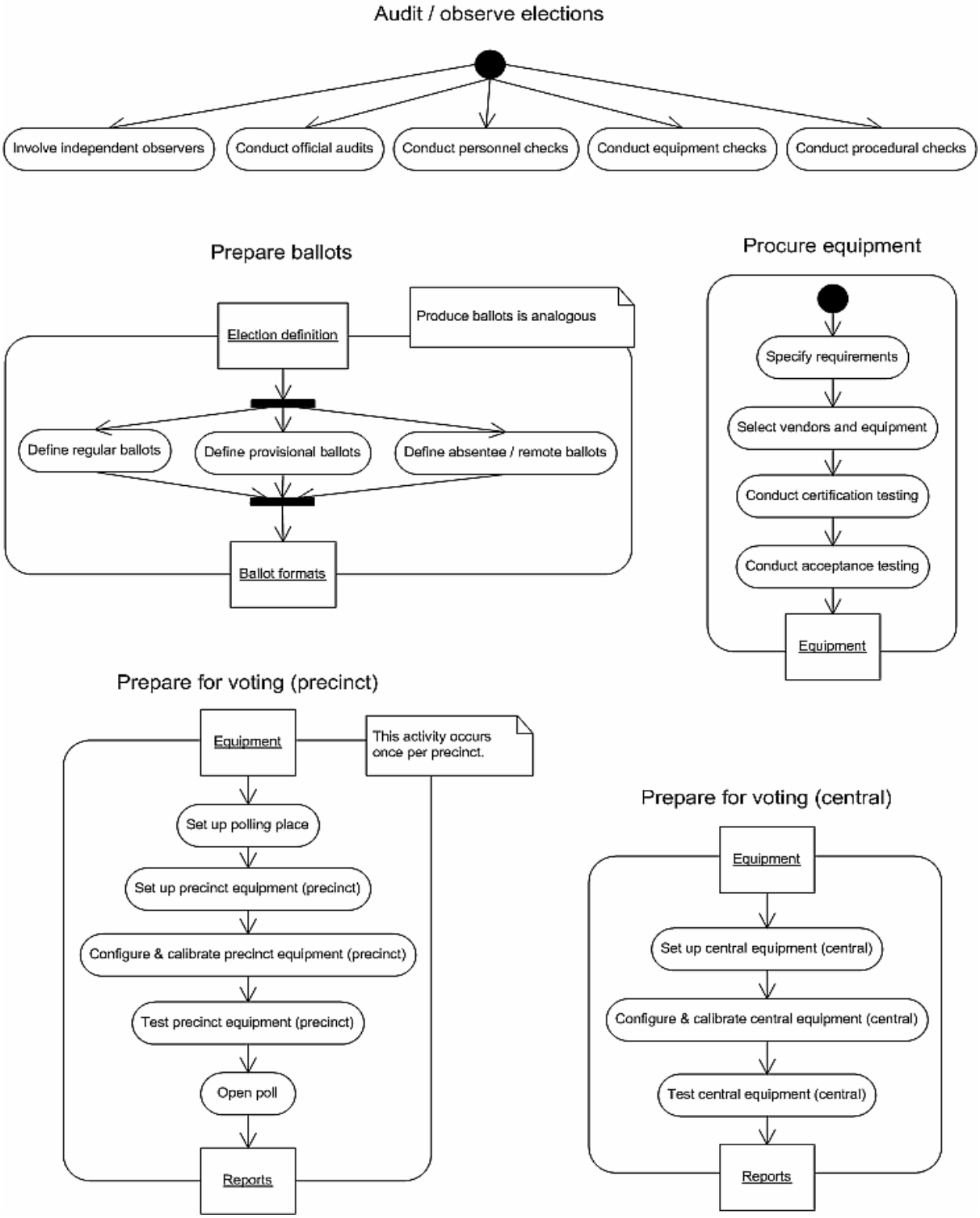

Figure 8 Miscellaneous activities (1)

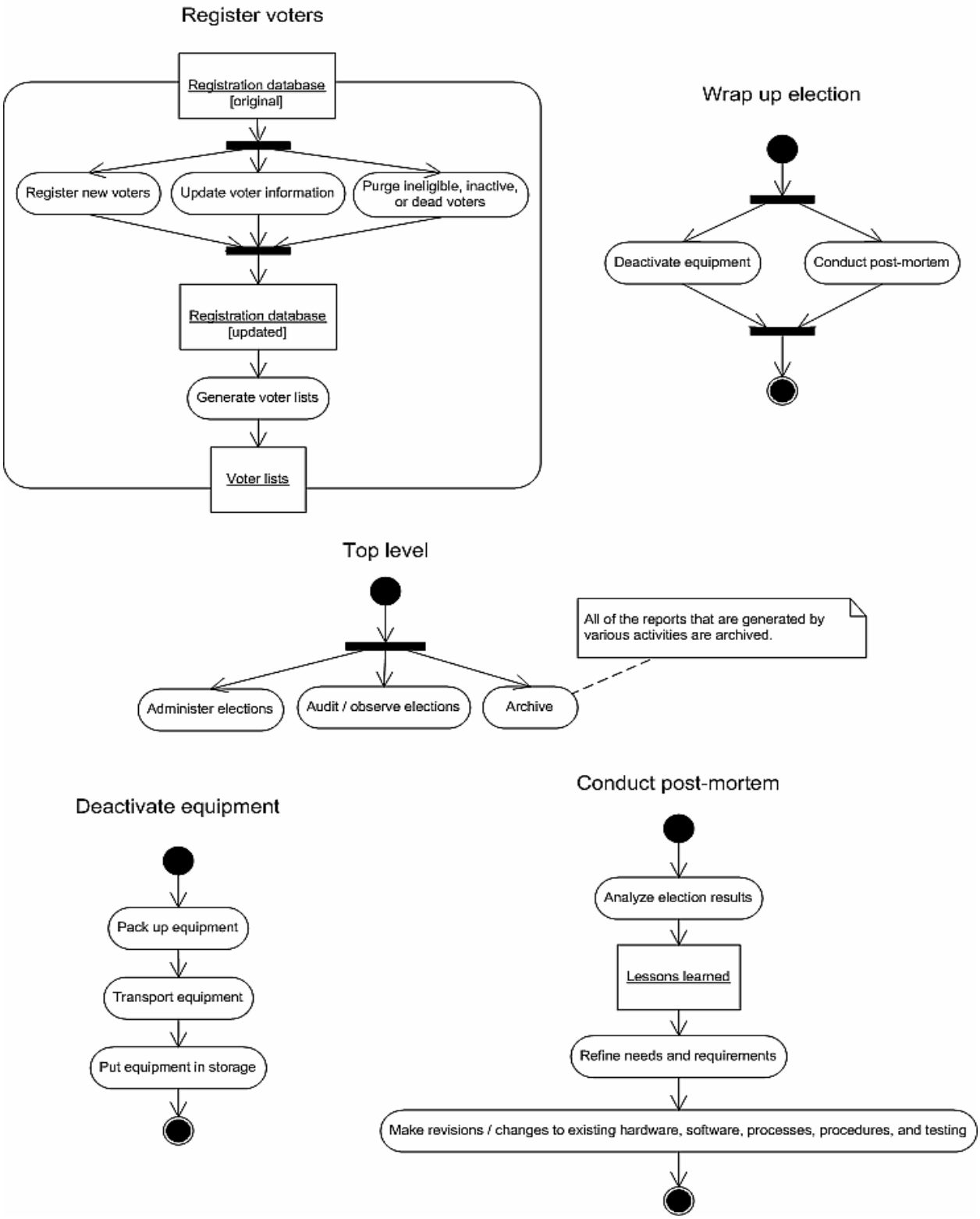

Figure 9 Miscellaneous activities (2)

Section 4.5.1.3 Translation of diagrams

This subsection contains a rendering of the process model into text. The rendering is based on Petri Net Linear Form [11].

Although the form of the diagrams is being changed from drawings to text, the meanings of the diagram elements -- activities, objects, etc. -- continue to be as in UML 2.0 [4].

Activities are represented in this translation by the activity name in parenthesis. Objects are represented in this translation by the object name in square brackets. Sometimes the names of activities and objects will themselves be qualified by parenthetical phrases or object states in square brackets. These have been retained as-is, nesting the parenthesis or brackets as needed.

Sequential control and object flows are indicated with ->.

A flow may be qualified by a guard condition and/or a multiplicity such as 0..1. These notations are inserted immediately before and after the affected flow. For example, Daytime->0..1(Drink coffee) denotes an optional flow into the "drink coffee" activity that can only occur if the condition Daytime is true.

A node may be assigned an identifier that may be used as the target of flows from elsewhere in the diagram. The identifier is prefixed by an asterisk and is introduced by including it after the first occurrence of the node name. For example, (Do something *s) denotes an activity "do something" with the identifier *s. The node name may be omitted in subsequent references that include only the identifier.

The following special nodes appear with semantics as in UML 2.0. They are distinguished from objects and activities by being enclosed between < and >.

- <InitialNode>
- <ForkNode>
- <JoinNode>
- <DecisionNode>
- <MergeNode>
- <ActivityFinal>
- <FlowFinal>

When multiple flows follow from a node, they are listed between curly braces {} and separated by commas.

A semicolon indicates that the description is about to continue at a different node. A period indicates that the description of the diagram is complete.

Translation of the diagrams follows.

```
Diagram:  Administer elections

<InitialNode>
```

```
  -><MergeNode *merge>
  ->(Prepare for election)
  ->[Equipment, voter lists, ballot formats and/or ballots]
  -><ForkNode>{
    ->(Prepare for voting (precinct))
      -><ForkNode>{
        ->(Gather in-person vote)
          ->[Ballots and/or ballot images]
          ->(Collect *c),
        Precinct count
          ->(Count (precinct count))
          ->[Machine totals]
          ->0..1(*c)
      },
    ->(Gather absentee / remote votes)
      ->[Ballots and/or ballot images]
      ->(*c),
    ->(Prepare for voting (central))
      ->(Wrap up voting (central) *w)
  };
(*c)
  ->[Ballots, ballot images and/or machine totals]
  ->(Wrap up voting (precinct))
  ->[Ballots, ballot images and/or precinct totals]
  ->(Wrap up voting (central) *w)
  ->[Counts [certified]]
  ->(Wrap up election)
  -><*merge>.

Note (on Gather in-person vote):  Includes early voting.


Diagram:  Prepare for election

Output:  [Equipment, voter lists, ballot formats and/or ballots]

<InitialNode>
  -><ForkNode>{
    ->(Define precincts)
      ->[Precinct definitions]
      -><ForkNode>{
        ->(Train poll workers)
          -><FlowFinal>,
        ->(Register voters)
          ->[Voter lists]
          ->(Collect *c1),
        ->(Program election)
          ->[Election definition]
          ->(Prepare ballots)
          ->[Ballot formats]
          -><ForkNode>{
            ->(*c1),
            Centrally programmed ballot formats
              ->[Ballot formats]
              ->0..1(Configure & calibrate precinct equipment (central) *cc)
          }
      },
```

```
    ->(Maintain equipment in storage)
      ->[Equipment [old]]
      ->(*cc),
    Need new equipment
      ->(Procure equipment)
      ->[Equipment [new]]
      ->0..1(*cc)
  };
(*c1)
  ->[Voter lists, ballot formats]
  -><ForkNode>{
    ->(Educate / notify / inform voters)
      -><FlowFinal>,
    ->(Collect *c2),
    Paper ballots
      ->(Produce ballots)
      ->[Ballots]
      ->0..1(*c2)
  };
(*cc)
  ->[Equipment [configured]]
  ->(Test precinct equipment (central))
  ->[Equipment [tested]]
  ->(Transport equipment)
  ->[Equipment [deployed]]
  ->(Collect *c2)
  ->[Equipment, voter lists, ballot formats and/or ballots].
```

Note (on Define precincts): This activity refers to configuring the voting system to realize the precincts as defined by state law.

Diagram: Gather in-person vote (paper-based).

This diagram is divided to show which activities are done by the voter and which are done by the poll worker. The activity Spoil ballot may be done by either. Present credentials, Mark ballot, Review ballot, and Present / submit ballot are done by the voter. All others are done by the poll worker.

Note: This activity occurs once per voter.

Input: [Voter lists]
Output: [Ballot [accepted]]

```
[Voter lists]
  ->(Check identity of voter *check);
<InitialNode>
  ->(Present credentials)
  ->(Check identity of voter *check)
  ->(Check voter eligibility)
  -><MergeNode *merge>
  ->(Update poll book)
  ->(Issue ballot or provisional ballot)
  ->(Provide private voting station)
  ->[Ballot [blank]]
  ->(Mark ballot)
```

```
  -><DecisionNode>{
    Fled voter
      ->(Spoil ballot)
      -><ActivityFinal>,
    else
      ->(Review ballot)
      -><DecisionNode>{
        Not OK
          ->(Spoil ballot)
          -><*merge>,
        OK
          ->(Present / submit ballot)
          ->[Ballot [completed]]
          ->(Validate ballot)
          -><DecisionNode>{
            OK
              ->(Accept ballot)
              ->[Ballot [accepted]],
            Not OK
              -><DecisionNode>{
                Try again
                  -><*merge>,
                else
                  -><ActivityFinal>
              }
          }
      }
  }.
```

Diagram: Gather in-person vote (DRE).

This diagram is divided to show which activities are done by the voter and which are done by the poll worker. The activity Spoil ballot may be done by either. Present credentials, Mark ballot, Review ballot, and Cast ballot are done by the voter. All others are done by the poll worker.

Note: This activity occurs once per voter.

Input: [Voter lists]
Output: [Ballot image]

```
[Voter lists]
  ->(Check identity of voter *check);
<InitialNode>
  ->(Present credentials)
  ->(Check identity of voter *check)
  ->(Check voter eligibility)
  ->(Update poll book)
  ->(Provide private voting station)
  -><MergeNode *merge>
  ->(Mark ballot)
  -><DecisionNode>{
    Fled voter
      ->(Spoil ballot)
      -><ActivityFinal>,
```

```
    else
      ->(Review ballot)
      -><DecisionNode>{
        Not OK
          ->(Spoil ballot)
          -><*merge>,
        OK
          ->(Cast ballot)
          ->[Ballot image]
      }
  }.
```

Diagram: Wrap up voting (precinct)

Note: This activity occurs once per precinct. Absentee / remote ballots may be handled and processed as a separate precinct under this activity.

Input: [Ballots, ballot images and/or machine totals]
Outputs: [Reports], [Ballots, ballot images and/or precinct totals [validated]]

```
[Ballots, ballot images and/or machine totals]
  ->(Close polls (including absentee / remote voting)){
    ->[Reports],
    ->[Ballots, ballot images and/or precinct totals [unvalidated]]
    -><MergeNode *merge>
    ->(Validate counts (precinct))
    -><DecisionNode>{
      Invalid
        ->(Diagnose and correct problem (precinct))
        ->[Ballots, ballot images and/or precinct totals [corrected,
unvalidated]]
        -><*merge>,
      else
        ->[Ballots, ballot images and/or precinct totals [validated]]
        ->(Deliver / transmit ballots, ballot images and/or precinct totals
to central)
        ->[Ballots, ballot images and/or precinct totals [validated]]
    }
  }.
```

Diagram: Wrap up voting (central)

Input: [Ballots, ballot images and/or precinct totals [validated]]
Outputs: [Counts [certified]], [Reports [official]]

```
[Ballots, ballot images and/or precinct totals [validated]]
  -><MergeNode *merge1>
  ->(Count (central))
  ->[Counts [unvalidated]]
  -><MergeNode *merge2>
  ->(Validate counts (central))
  -><DecisionNode>{
    Invalid
```

```
      ->(Diagnose and correct problem (central))
      ->[Counts [corrected, unvalidated]]
      -><*merge2>,
    else
      ->[Counts [validated]]
      ->(Generate unofficial reports)
      ->[Reports [unofficial]]
      ->(Reconcile provisional/challenged ballots)
      ->[Counts [adjusted]]
      ->(Generate official reports)
      ->[Reports [official]]
      -><DecisionNode>{
        Recount
          ->(Retrieve original data)
          ->[Ballots, ballot images and/or precinct totals [validated]]
          -><*merge1>,
        else
          ->(Certify final counts){
            ->[Counts [certified]],
            ->[Reports [official]]
          }
      }
  }.

Note (on Count (central)):  Including absentee and write-ins.


Diagram:  Audit / observe elections

<InitialNode>{
  ->(Involve independent observers),
  ->(Conduct official audits),
  ->(Conduct personnel checks),
  ->(Conduct equipment checks),
  ->(Conduct procedural checks)
}.


Diagram:  Prepare ballots

Note:  Produce ballots is analogous.

Input:  [Election definition]
Output:  [Ballot formats]

[Election definition]
  -><ForkNode>{
    ->(Define regular ballots)
      -><JoinNode *j>,
    ->(Define provisional ballots)
      -><*j>,
    ->(Define absentee / remote ballots)
      -><*j>
  };
<*j>
  ->[Ballot formats].
```

```
Diagram:  Procure equipment

Output:  [Equipment]

<InitialNode>
  ->(Specify requirements)
  ->(Select vendors and equipment)
  ->(Conduct certification testing)
  ->(Conduct acceptance testing)
  ->[Equipment].


Diagram:  Prepare for voting (precinct)

Note:  This activity occurs once per precinct.

Input:  [Equipment]
Output:  [Reports]

[Equipment]
  ->(Set up polling place)
  ->(Set up precinct equipment (precinct))
  ->(Configure & calibrate precinct equipment (precinct))
  ->(Test precinct equipment (precinct))
  ->(Open poll)
  ->[Reports].


Diagram:  Prepare for voting (central)

Input:  [Equipment]
Output:  [Reports]

[Equipment]
  ->(Set up central equipment (central))
  ->(Configure & calibrate central equipment (central))
  ->(Test central equipment (central))
  ->[Reports].


Diagram:  Register voters

Input:  [Registration database [original]]
Output:  [Voter lists]

[Registration database [original]]
  -><ForkNode>{
    ->(Register new voters)
      -><JoinNode *j>,
    ->(Update voter information)
      -><*j>,
    ->(Purge ineligible, inactive, or dead voters)
      -><*j>
  };
<*j>
  ->[Registration database [updated]]
```

```
  ->(Generate voter lists)
  ->[Voter lists].


Diagram:  Wrap up election

<InitialNode>
  -><ForkNode>{
    ->(Deactivate equipment)
      -><JoinNode *j>,
    ->(Conduct post-mortem)
      -><*j>
  };
<*j>
  -><ActivityFinal>.


Diagram:  Top level

<InitialNode>
  -><ForkNode>{
    ->(Administer elections),
    ->(Audit / observe elections),
    ->(Archive)
  }.

Note (on Archive):  All of the reports that are generated by various
activities are archived.


Diagram:  Deactivate equipment

<InitialNode>
  ->(Pack up equipment)
  ->(Transport equipment)
  ->(Put equipment in storage)
  -><ActivityFinal>.


Diagram:  Conduct post-mortem

<InitialNode>
  ->(Analyze election results)
  ->[Lessons learned]
  ->(Refine needs and requirements)
  ->(Make revisions / changes to existing hardware, software, processes,
procedures, and testing)
  -><ActivityFinal>.
```

Section 4.5.2 Logic model (normative)

This model defines the results that must appear in vote data reports and is used in verification of voting system logic. It does not address ranked order voting and does not attempt to define every voting variation that jurisdictions may use. It suffices for N of M (including 1 of M) and

cumulative voting.

Section 4.5.2.1 Domain of discourse

Term	Definition
A(*t*,*v*)	Boolean function, returns true if and only if voter *v*'s ballot or ballot image conforms to jurisdiction-dependent criteria for accepting or rejecting entire ballots, such as stray marks policies and voter eligibility criteria, as of time *t*. This value is false for provisional, challenged, and review-required ballots that are not [yet] validated. The system may not be able to determine the value of A(*t*,*v*) without human input; however, it may assign tentative values according to local procedures and state law, to be corrected later if necessary by input from election workers. The value of A(*t*,*v*) may change over time as a result of court decisions, registrar review of voter eligibility, etc. In a paper-based system, A(*t*,*v*) will be false if voter *v*'s ballot is unprocessable.
B(*v*)	The time at which voter *v* begins voting (i.e., when the ballot is enabled; the start of the voting session).
C(*r*)	The set of all candidates or choices that are "on the ballot" in a contest *r*. Write-in candidates do not appear in C(*r*). *
C'(*r*,*t*)	The set of all candidates or choices for a contest *r*, including any write-ins that the voters have written in as of time *t*. Each distinct write-in candidate appears separately in C'(*r*,*t*). Where write-ins are not allowed, C'(*r*,*t*) = C(*r*). *
c, c_n, etc.	Individual candidates or choices.
D(*v*)	The time at which voter *v* is done voting (the time at which the ballot is cast or the ballot of a fled voter is spoiled; the end of the voting session).
J	The set of reporting contexts (including tabulators, precincts, election districts, and jurisdiction).
j, j_n, etc.	Individual reporting contexts.

$K(j,r,t)$	For a given contest and reporting context, the number of read ballots for which $A(t,v)$ is true as of time t (i.e., the number of ballots that should be counted). Ballot formats that do not include contest r do not contribute to this total.
L_B	A limit on the number of ballots or ballot images that the system is claimed to be capable of processing correctly.
L_C	A limit on the number of ballot positions per contest that the system is claimed to be capable of processing correctly. (See also L_W)
L_F	A limit on the number of ballot formats that the system is claimed to be capable of processing correctly.
L_R	A limit on the number of contests that the system is claimed to be capable of processing correctly.
L_T	A numerical limit on vote totals that the system is claimed to be capable of processing correctly.
L_V	A limit on the number of voters casting provisional, challenged, or review-required ballots that the system is claimed to be capable of processing correctly.
L_W	A limit on the total number of distinct candidates or choices per contest, including write-ins, that the system is claimed to be capable of processing correctly. It shall be that $L_W \geq L_C$. (See also L_C)
$N(r)$	The maximum number of votes that may be cast by a given voter in contest r, pursuant to the definition of the contest. For N of M contests, this is the value N.
$O(j,r,t)$	For a given contest and reporting context, the number of overvotes in read ballots for which $A(t,v)$ is true as of time t. Each ballot in which contest r is overvoted contributes $N(r)$ to $O(j,r,t)$.
R	The set of all contests.
r, r_n, etc.	Individual contests in R.
$S(c,r,t,v)$	Voter v's vote with respect to candidate or choice c in contest r as of time t. For checkboxes and the like, the value shall be 1 (selected) or 0 (not selected). For cumulative voting, the value shall be the number of votes that v gives to candidate or choice c in contest r. If the applicable ballot format does not include contest r, $S(c,r,t,v) = 0$.

S'(*c*,*r*,*t*, *v*)	Voter *v*'s vote with respect to candidate or choice *c* in contest *r* as accepted for counting purposes (i.e., valid votes only), as of time *t*.
S(*r*,*t*,*v*)	The total number of votes that voter *v* has cast in contest *r* as of time *t*. $S(r,t,v) = \sum_{c \in C'(r,t)} S(c,r,t,v)$
T(*c*,*j*,*r*,*t*)	The vote total for candidate or choice *c* in contest *r* and reporting context *j* as of time *t*. This does not include votes that are invalid due to overvoting or votes from ballots for which A(*t*,*v*) is false.
t, t_n, etc.	Individual time points.
t_O	The time at which polls are opened.
t_C	The time at which polls are closed.
t_E	The time at which the value of A(*t*,*v*) is frozen for all voters, the counting is complete, and final vote totals are required ("end").
U(*j*,*r*,*t*)	For a given contest and reporting context, the number of undervotes in read ballots for which A(*t*,*v*) is true as of time *t*. A given ballot contributes at most N(*r*) to U(*j*,*r*,*t*). Ballot formats that do not include contest *r* do not contribute to this total.
V(*j*,*t*)	The set of all voters within reporting context *j* who have begun voting by time *t*, including any voter that is presently voting.
v, v_n, etc.	Individual voters in V(*j*,*t*).

Replace "voters" with "ballots" as much as possible: (A) avoid suggesting loss of privacy; (B) systems do not identify voters; one-voter-one-ballot is enforced by poll workers.

* The fact that some systems initially report "Write-In" as a single ballot position, leaving the distribution of votes to different write-in candidates for post-processing, is an implementation detail. These standards contain requirements on the information content of the final report, which must provide separate totals for each write-in candidate.

ISSUE: voting processes with manual / optional processing of ballots with write-in votes. The inclusion of votes from write-in ballots may be outside the system and unverifiable. Can a system within such a process possibly conform to the write-ins profile? See also Requirement 4.4.3.2-1.3.

Section 4.5.2.2 General assertions

Invariants:

$$t_O < t_C \leq t_E$$

$$v \in V(j,t) \rightarrow B(v) \leq t$$

$$B(v) < D(v)$$

$$S(c,r,t,v) \geq 0$$

$$S'(c,r,t,v) \geq 0$$

$$S(c,r,t,v) > 0 \rightarrow c \in C'(r,t)$$

The following assertions formalize a subset of the compliance points appearing in Section 4.3. Each textual assertion is intended to elucidate the formal assertion(s) that follow it. In case of discrepancy or confusion, the formal assertions are normative.

No one shall vote before polls are opened or after polls have closed, or during the process of opening or closing the polls.

$$B(v) > t_O$$

$$D(v) < t_C$$

A voter shall have no votes before he or she begins voting.

$$t < B(v) \rightarrow S(r,t,v) = 0$$

A voter's votes shall not change once the voter is done voting.

$$t \geq D(v) \rightarrow S(c,r,t,v) = S(c,r,D(v),v)$$

Section 4.5.2.3 Cumulative voting

All valid votes shall be counted.[10]

$$t \geq D(v) \wedge S(r,D(v),v) \leq N(r) \wedge A(t,v) \rightarrow S'(c,r,t,v) = S(c,r,D(v),v)$$

No invalid votes shall be counted.

$$t \geq D(v) \wedge \left(S(r,D(v),v) > N(r) \vee \overline{A(t,v)}\right) \rightarrow S'(c,r,t,v) = 0$$

The final vote totals shall accurately reflect all valid votes and only valid votes.

$$T(c,j,r,t_E) = \sum_{v \in V(j,t_E)} S'(c,r,t_E,v)$$

Every vote shall be accounted for.

$$\sum_{c \in C'(r,t_E)} T(c,j,r,t_E) + O(j,r,t_E) + U(j,r,t_E) = K(j,r,t_E) \times N(r)$$

Section 4.5.2.4 N of M contests (including 1-of-M)

N of M is identical to cumulative voting but for the addition of the following invariant, which reflects the design of a ballot format that allows only one vote in each ballot position (equivalent to a checkbox).

$$S(c,r,t,v) \leq 1$$

Section 4.6 Best practices for election officials

Requirement 4.6-1 Election officials shall verify that paper ballots are produced in accordance with vendor specifications.

Requirement 4.6-2 All printed copy records produced by the election database and ballot processing systems shall be labeled and archived for a period of at least 22 months after the election.

Origin: Reworded from [1] I.2.2.11.

Requirement 4.6-3 The voting process shall prevent a voter from voting on a ballot to which he or she is not entitled.

Origin: [1] I.2.4.2.c, generalized from DRE systems to the voting process.

Discussion: In practice, as of 2005, this requirement is managed by poll workers. However, the voting system may support this requirement.

Impact: [1] restriction to DRE was probably to require DREs to do this automatically, which they don't.

Requirement 4.6-4 The voting process shall prevent a voter from casting more than one ballot in the same election.

Origin: [1] I.2.4.2.d, generalized from DRE systems to the voting process.

Discussion: In practice, as of 2005, this requirement is managed by poll workers. However, the voting system may support this requirement.

Impact: [1] restriction to DRE was probably to require DREs to do this automatically, which they don't.

Requirement 4.6-5 The voting process shall prevent modification of the voter's vote after the ballot is cast.

Origin: [1] I.2.4.3.3.n, generalized. See also Requirement 4.4.2-4.

Discussion: See casting.

Requirement 4.6-6 The voting process shall prevent access to voted ballots until after the close of polls.

Origin: [1] I.2.4.3.3.r, generalized. See also Requirement 4.4.3.1-1.

Section 5 Standards on data to be provided

Section 5.1 Technical data package (vendor)

TDP stuff from [1] needs to be incorporated and cleaned up.

Section 5.1.1 Implementation statement

As define in Section 4.2.1.

Section 5.1.2 Source code and logical designs

(Changes / additions to current spec)

- Source code, for systems using software; analogous formal logic designs, for systems not using software.
- For systems using interpreted code, a specification of the version of the industry standard runtime interpreter that shall be used to run this code. (See Requirement 4.3.1.1.1-4.)
- For each callable unit (function, method, operation, subroutine, procedure, etc.) in source code or analogous logic design:
    - The preconditions and postconditions, formally stated using the terms defined in Section 4.5.2.1, including any assumptions about capacities and limits within which the system is expected to operate.[11]
    - A convincing argument (possibly, but not necessarily, a formal proof) that the

preconditions and postconditions accurately represent the behavior of the callable unit.[12] This requirement may be waived in specific cases where the VSTL agrees that it is obvious.

- A formal proof, using the preconditions and postconditions, that the software or logic design as a whole satisfies each of the invariants and assertions indicated in Section 4.5.2 for the profiles claimed in the implementation statement, for all cases within the aforementioned capacities and limits.

Postconditions that impact something outside the domain of discourse are not of interest unless that thing impacts the behavior of some function with respect to the domain of discourse. The vendor shall define such terms as are necessary to state any and all dependencies and assumptions that may impact the behavior of some function with respect to the domain of discourse and use them consistently in all affected preconditions and postconditions. *An excess of extraneous dependencies may negatively impact the VSTL's ability to verify the system's correctness and thereby prevent qualification.*

A callable unit might have no impact on anything in the domain of discourse and no dependency on anything in the domain of discourse. Such a unit shall have a true precondition and a postcondition that states that nothing in the domain of discourse is changed.

If a VSTL requests it, the vendor shall furnish proof that the selected coding conventions are published and credible.

Section 5.2 Voting equipment user documentation (vendor)

Section 5.2.1 Security-related procedures that the vendor must document for users

Requirement 5.2.1-1 For systems conforming to the *Central count* profile, the Voting Equipment User Documentation shall detail the measures to be taken related to the physical and procedural controls for handling of ballot boxes.

Origin: Reworded from [1] I.6.3.2

Test reference: Section 6.2.1

Requirement 5.2.1-2 For systems conforming to the *Central count* profile, the Voting Equipment User Documentation shall detail the measures to be taken related to the physical and procedural controls for preparing of ballots for counting.

Origin: Reworded from [1] I.6.3.2

Test reference: Section 6.2.1

Requirement 5.2.1-3 For systems conforming to the *Central count* profile, the

Voting Equipment User Documentation shall detail the measures to be taken related to the physical and procedural controls for counting operations.

Origin: Reworded from [1] I.6.3.2

Test reference: Section 6.2.1

Requirement 5.2.1-4 For systems conforming to the *Central count* profile, the Voting Equipment User Documentation shall detail the measures to be taken related to the physical and procedural controls for reporting data.

Origin: Reworded from [1] I.6.3.2

Test reference: Section 6.2.1

Section 5.2.2 Approved parts list

Requirement 5.2.2-1 For marking devices manufactured by multiple external sources, the vendor shall provide a listing of sources and model numbers that are compatible with the system.

Origin: [1] I.3.2.4.2.1.c; also II.2.9.4.2.a

Requirement 5.2.2-2 The user documentation shall specify the required paper stock, size, shape, opacity, color, watermarks, field layout, orientation, size and style of printing, size and location of punch or mark fields used for vote response fields and to identify unique ballot formats, placement of alignment marks, ink for printing, and folding and bleed-through limitations for preparation of ballots that are compatible with the system.

Origin: [1] I.3.2.4.2.1.c; also II.2.9.4.2.a

Section 5.2.3 Availability analysis

Vendors shall specify the typical system configuration that is to be used to assess availability, and any assumptions made with regard to any parameters that impact the MTTR. These factors shall include at a minimum:

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

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

c. Organizational affiliation (i.e., jurisdiction, vendor) of qualified maintenance

personnel

Section 5.2.4 Etc.

Tons of stuff in [1] II.2; search [1] and [2] for applicable requirements.

> Requirement 5.2.4-1 The user documentation supplied by the vendor shall include a statement of all requirements and restrictions regarding environmental protection, electrical service, recommended auxiliary power, telecommunications service, and any other facility or resource required for the proper installation and operation of the system.
>
> Impact: From VVSG2-20050706-ag. Changed TDP to user documentation.

Section 5.3 Test report for EAC certification (VSTL)

The term "finding" refers to a result of the VSTL's formal inquiry (a verdict).[13]

Whether or not a system is qualified, the VSTL shall report all of the data collected for estimation of MTBF and error rate.

If a system is not qualified, the VSTL shall report on all failed tests and the reasons for failure, including all applicable evidence (e.g., vote data report, citation of logic error in source code).

The VSTL shall report the implementation-dependent structural tests performed and the test verdicts. No system shall be qualified if any implementation-dependent structural tests are assigned the verdict Fail using the VSTL's defined pass criteria.

The VSTL shall report the implementation-dependent functional tests performed and the test verdicts. No system shall be qualified if any implementation-dependent functional tests are assigned the verdict Fail using the VSTL's defined pass criteria.

For each callable unit (function, method, operation, subroutine, procedure, etc.), in source code or analogous logic design, the VSTL shall report a finding on whether the preconditions and postconditions correctly describe the behavior of the unit in all cases. This finding shall be one of Correct, Incorrect, or Unable to Determine.

The VSTL shall report a finding whether all the assumptions about capacities and limits that appear in the preconditions, postconditions, and proofs are consistent with the capacities and limits that the system is claimed to be capable of processing correctly. This finding shall be one of Consistent, Inconsistent, or Unable to Determine.

For the software or logic design as a whole, and for each invariant and assertion indicated in Section 4.5.2 for the profiles claimed in the implementation statement, the VSTL shall report a finding whether the assertion is satisfied in all cases within the aforementioned capacities and

limits. This finding shall be one of Satisfied, Unsatisfied, or Unable to Determine.

STS, HFP: add your stuff.

Section 5.4 Public Information Package (VSTL)

If a system is qualified, the VSTL shall publish a statement to that effect that includes the following information:

- the implementation statement, as made by the vendor;
- a list of the tests for which the test verdict was Waived;
- the estimated error rate and MTBF of the system as calculated from the statistics collected during testing;
- for systems conforming to the *Marksense* or *Punchcard* profiles, the speed or rate at which tabulation was performed in typical case and capacity tests;
- a confirmation that the verdict on every applicable test was Pass, all preconditions and postconditions were found Correct, all the assumptions about capacities and limits were found Consistent, and all assertions were found Satisfied;
- a summary of all uncorrected nonconformities and anomalies that were noted during conformity assessment, no matter how minor;
- a timeline of the qualification process for the qualified system.

STS, HFP: add your stuff.

Section 6 Testing standard

Section 6.1 Overview of qualification testing

[1] defines qualification testing as "the examination and testing of a computerized voting system by an Independent Test Authority using qualification test standards to determine if the system complies with the qualification performance and test standards and with its own specifications. This process occurs prior to state certification."

The purpose of voting system (qualification) testing is to provide the states and other affected stakeholders with some level of assurance that a voting system is fit for use. States have the option to subject a voting system to additional scrutiny before purchasing and deploying it; however, most states require qualification by an ITA as an entry condition.

Even if procedural controls and audit trails ensured that any miscount would be detected, it could still be catastrophic for a state to have to rerun a compromised election and to remedy the faulty equipment. It is in the states' interests for the qualification process to eliminate voting systems that are not trustworthy before they are purchased and deployed.

Section 6.2 Introduction to test methods

Section 6.2.1 Inspection

Section 6.2.2 Expert review

Section 6.2.3 General functional testing

Section 6.2.4 General performance testing (benchmarking)

Section 6.2.4.1 Decibels, contrast ratios, response time, throughput, that kind of thing

Section 6.2.4.2 Tests with human subjects

Section 6.2.5 Software and logic testing

Overlap with logic verification rationale section in Overview (need the background to explain the rationale)

Section 6.2.5.1 Black box (insofar as black-box software and logic testing differs from or specializes general functional testing)

Section 6.2.5.2 White box

Section 6.2.5.2.1 Implementation-dependent structural tests

The VSTL shall review the vendor's program analysis, documentation, and, if available, module test case design. The VSTL shall evaluate the test cases for each module, with respect to flow control parameters and data on both entry and exit. All discrepancies between the Software Specifications and the test case design shall be corrected by the vendor prior to initiation of the qualification test.

If the vendor's module test case design does not provide conclusive coverage of all program paths, then the VSTL shall perform an independent analysis to assess the frequency and consequence of error of the untested paths. The VSTL shall define and execute additional module test cases as required to provide coverage of all modules containing untested paths with potential for untrapped errors. The VSTL shall define pass criteria for implementation-

dependent structural tests using the VVSG and the vendor-supplied system documentation to determine acceptable ranges of performance.

This text is retained from [1] II.A.4.3.3, "Software Module Test Case Design and Data," with minor changes. As time permits, this section should be rewritten to enhance repeatability and reproducibility of the testing.

Section 6.2.5.2.2 Implementation-dependent functional tests

The VSTL shall review the vendor's functional test case designs. The VSTL shall prepare a detailed matrix of system functions and the test cases that exercise them. The VSTL shall also prepare a test procedure describing all test ballots, operator procedures, and the data content of output reports. Abnormal input data and operator actions shall be defined. Test cases shall also be designed to verify that the system is able to handle and recover from these abnormal conditions.

The vendor's test case design may be evaluated by any standard or special method appropriate.

In the event that the vendor's functional test data are insufficient, the VSTL shall define and execute additional functional tests. The VSTL shall define pass criteria for implementation-dependent functional tests using the VVSG and the vendor-supplied system documentation to determine acceptable ranges of performance.

Depending upon the design and intended use of the voting system, all or part of the functions listed below shall be tested.

a. Ballot preparation subsystem;

b. Test operations performed prior to, during, and after processing of ballots, including:

1. Logic tests to verify interpretation of ballot styles, and recognition of precincts to be processed;
2. Accuracy tests to verify ballot reading accuracy;
3. Status tests to verify equipment statement and memory contents;
4. Report generation to produce test output data; and
5. Report generation to produce audit data records;

c. Procedures applicable to equipment used in the polling place for:

1. Opening the polling place and enabling the acceptance of ballots;
2. Maintaining a count of processed ballots;
3. Monitoring equipment status;
4. Verifying equipment response to operator input commands;
5. Generating real-time audit messages;
6. Closing the polling place and disabling the acceptance of ballots;
7. Generating election data reports;

8. Transfer of ballot counting equipment, or a detachable memory module, to a central counting location; and
9. Electronic transmission of election data to a central counting location; and

d. Procedures applicable to equipment used in a central counting place:

1. Initiating the processing of a ballot deck, programmable memory device, or other applicable media for one or more precincts;
2. Monitoring equipment status;
3. Verifying equipment response to operator input commands;
4. Verifying interaction with peripheral equipment, or other data processing systems;
5. Generating real-time audit messages;
6. Generating precinct-level election data reports;
7. Generating summary election data reports;
8. Transfer of a detachable memory module to other processing equipment;
9. Electronic transmission of data to other processing equipment; and
10. Producing output data for interrogation by external display devices.

This text is retained from [1] II.A.4.3.4, "Software Functional Test Case Design," with minor changes. As time permits, this section should be rewritten to enhance repeatability and reproducibility of the testing.

Section 6.3 Documentation and design reviews (expert review)

Section 6.3.1 Logic verification

Because of its high complexity, the scope of logic verification is necessarily limited to the core vote gathering and tabulating functions of specific components of the voting system (a voting machine and/or a central tabulator).

(Paste in standard text on logic verification.)

No system shall be qualified unless all preconditions and postconditions (see Section 5.1.2 and Section 5.3) are found Correct.

No system shall be qualified unless all the assumptions about capacities and limits (see Section 4.2.1 and Section 5.3) are found Consistent.

No system shall be qualified unless all invariants and assertions (see Section 5.1.2 and Section 5.3) are found Satisfied.

Section 6.3.2 Verification of design requirements in product standard

For each of the design requirements enumerated below, the VSTL shall review the source code (if applicable) and design of the voting system to verify that the requirement is satisfied. For each one, the VSTL shall publish a finding whether the requirement is met. This finding shall be one of Satisfied, Unsatisfied, or Unable to Determine. No system shall be qualified unless all design requirements are found Satisfied.

- Requirement 4.3.5.1-1[14]
- Requirement 4.3.6-1[15]
- Requirement 4.4.2-2.15
- Requirement 4.4.3.3-18
- Requirement 4.4.3.4-1
- Requirement 4.4.3.4-2.1
- Requirement 4.4.3.4-2.2

Complete this list

Section 6.3.3 *Et seq.* Stuff retained from [1] Vol. II

Section 6.3.3.1 Source code review [v2s5 20050301 5.4]

Overlap with logic verification

This pertains to the callable unit length limit requirement:

> Requirement 6.3.3.1-1 The reviewer should consider the functional organization of the module [sic] and the use of formatting, such as blocking into readable units, which supports the intent of this requirement where the module itself, excluding comments, exceeds the limits. The vendor shall justify, to the satisfaction of the VSTL, any module lengths exceeding this objective.[166]
>
> Origin: [1] II.5.4.2.i, as revised by Section 6.6.4.2, Paragraph i of [5].
>
> Move the justification-to-VSTL to the TDP?

Requirements regarding code generators. There are COTS code generators and there are non-COTS ones. Ramifications are different.

Section 6.4 Test protocols

Section 6.4.1 Counting and reporting

Section 6.4.1.1 General test template

Most test cases will follow this general template. Different test cases will elaborate on the general template in different ways, depending on what is being tested.

a. Establish initial state (clean out data from previous tests, verify resident software/firmware)

b. Program election and prepare ballots

c. Generate pre-election audit reports

d. Configure polling equipment

e. Generate system readiness audit reports

f. Open poll

g. Run test ballots

h. Close poll

i. Generate in-process audit reports

j. Generate data reports for the specified reporting contexts

k. Inspect ballot counters

l. Inspect reports

Section 6.4.1.2 General pass criteria

The VSTL need only consider tests that are applicable to the profiles claimed in the implementation statement and those tests that are designated for all systems. The test verdict for all other tests shall be Not Applicable.

If the documented assumptions for a given test (indicated by the presence of an **Assumptions:** field in the test case description) are not met, the test verdict shall be Waived and the test shall not be executed.

If the VSTL is unable to execute a given test because the system does not support functionality that is required per the implementation statement or is required for all systems, the test verdict shall be Fail.

If the VSTL executes a test, the test verdict shall be assigned based on the following inputs, which are described in more detail below:

- Mean Time Between Failure (MTBF)
- Error rate
- Additional pass criteria
- General performance requirements

The test verdict shall be Pass if and only if none of these inputs indicates a verdict of Fail.

No system shall be qualified if any test verdicts are Fail.

Section 6.4.1.2.1 Mean Time Between Failure

During execution of all tests except Dangling ref: ForcedErrorRecovery, the VSTL shall keep track of real time and the number of failures (see definition in Section 3). These statistics shall be collected and accumulated across all tests.

If a failure should occur during the execution of any test except Dangling ref: ForcedErrorRecovery, the VSTL shall note the failure for use in the calculation of MTBF. The VSTL shall then follow the vendor's documented procedures for recovering from failures. If recovery is not possible or not successful, the test verdict shall be Fail. Otherwise, after recovery, the VSTL shall attempt to re-execute the test that was affected by the failure from the beginning. If the failure reoccurs, the test verdict shall be Fail. If the failure does not reoccur, the following system-level MTBF decision criteria shall be applied:

> If statistical analysis of the cumulative behavior across all tests executed so far indicates with at least 90% confidence that the MTBF is worse than XXX hours, the test verdict shall be Fail.
>
> Otherwise, the failure shall be noted, the test verdict shall be assigned based on the other inputs (disregarding the failure), and testing shall continue.

The MTBF issue. Currently, [3] II.C.4 only establishes 90% confidence that MTBF exceeds 45 hours.

Section 6.4.1.2.2 Error rate

During all test executions, the VSTL shall keep track of the number of ballot positions counted and the number of errors (see definition in Section 3). These statistics shall be collected and accumulated across all tests.

If a test runs to completion, the VSTL shall inspect the data reports and verify that counts and totals are reported in compliance with the requirements in Section 4.4.3.3. If all reported counts and totals are identical to the specified values, the test verdict shall be Pass. Otherwise, the following system-level accuracy decision criteria shall be applied:

> If statistical analysis of the cumulative behavior across all tests executed so far indicates with at least 95% confidence that the error rate is worse than 1 in 10,000,000 ballot positions, the test verdict shall be Fail.
>
> Otherwise, the error shall be noted, the test verdict shall be assigned based on the other inputs (disregarding the error), and testing shall continue.

Section 6.4.1.2.3 Additional pass criteria

When certain performance requirements of the VVSG are of particular relevance to a particular test, these are noted after **Additional pass criteria:** in the test case description. The VSTL shall verify that these requirements are met during the execution of that test case; if they are not, the test verdict shall be Fail.

Section 6.4.1.2.4 General performance requirements

A demonstrable violation of any requirement of the VVSG during the execution of any test case shall result in a test verdict of Fail, irrespective of whether this requirement was explicitly noted in the Additional pass criteria for that test case.

For example, if any of the audit reports should be incomplete or incorrect with respect to any of the many applicable requirements in Section 4.3.1.2, the test verdict would be Fail.

For example, if a DRE system should take longer than 5 minutes for each device to generate a consolidated report, Requirement 4.4.3.1-5 would be violated and the test verdict would be Fail.

Section 6.4.1.3 Null case tests

The purpose of the null case test is to verify that closing the polls after processing zero ballots is correctly handled. This case can arise in practice, for example, in precincts where a single DRE is provided alongside other equipment, if no voters use the DRE.

Section 6.4.1.3.1 All systems

Test 1 Null Case

> **References:** Requirement 4.4.3.3-1.2

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Null Case Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

> Unopposed Candidate

Reporting contexts: Single context.

Scenario:

No ballots shall be cast.

Section 6.4.1.4 Functional tests

The purpose of a functional test is to establish that one or more functional features that are required to be supported, are supported. Functional tests are not stress tests, although by their minimality, they may unintentionally test boundary conditions. For stress tests, refer to the Capacity Tests section.

Following subsections are organized by compliance profiles. Functional tests are applicable only if the implementation statement claims the profile indicated in the subsection name.

Section 6.4.1.4.1 All systems

Test 2 1-of-M Trivial Case

References: Requirement 4.4.1.1-2.1.1, Requirement 4.4.1.1-2.1.2, Requirement 4.4.2-2.3, Requirement 4.4.3.2-1.1

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the 1-of-M Trivial Case Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

Unopposed Candidate

Reporting contexts: Single context.

Scenario:

Two ballots shall vote for Unopposed Candidate.

Test 3 1-of-M Simple Case

References: Requirement 4.4.1.1-2.1.1, Requirement 4.4.1.1-2.1.2, Requirement 4.4.2-2.3, Requirement 4.4.3.2-1.1

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

This is the only contest in the 1-of-M Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

Ballot Position 1
Ballot Position 2
Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote for Ballot Position 1

Three ballots shall vote for Ballot Position 2

Two ballots shall vote for Ballot Position 3

One ballot shall vote for none (undervote).

Test 4 Reporting Levels Test

References: Requirement 4.4.1.1-1

Six voting machines, three precinct tabulators and one jurisdiction tabulator are required to execute this test.

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Reporting Levels Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts:

Machines 1 and 2 shall be in Precinct 1.

Machines 3 and 4 shall be in Precinct 2.

Machines 5 and 6 shall be in Precinct 3.

Precincts 1 and 2 shall be in District 1.

Precinct 3 shall be in District 2.

All of the above shall be in the Jurisdiction.

Scenario:

On Machine 1, three ballots shall be cast for Ballot Position 1, two ballots shall be cast for Ballot Position 2, and one ballot shall be cast for Ballot Position 3.

On Machine 2, three ballots shall be cast for Ballot Position 1, one ballot shall be cast for Ballot Position 2, and one ballot shall be cast for Ballot Position 3.

On Machine 3, two ballots shall be cast for Ballot Position 1, one ballots shall be cast for Ballot Position 2, and one ballot shall be cast for Ballot Position 3.

On Machine 4, one ballot shall be cast for Ballot Position 1, one ballot shall be cast for Ballot Position 2, and one ballot shall be cast for Ballot Position 3.

On Machine 5, two ballots shall be cast for Ballot Position 2.

On Machine 6, one ballot shall be cast for Ballot Position 1.

Section 6.4.1.4.2 DRE

Test 5 Ballot Images Simple Case

References: Requirement 4.4.3.5-1.2, Requirement 4.4.3.5-1.3

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Ballot Images Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote for Ballot Position 1

Three ballots shall vote for Ballot Position 2

Two ballots shall vote for Ballot Position 3

One ballot shall vote for none (undervote).

After close of polls, the VSTL shall retrieve and review the ballot images.

Additional pass criteria:

The Cast Vote Records (retrieved ballot images) shall be accurate. (Requirement 4.4.3.5-1.2)

The Cast Vote Records (retrieved ballot images) shall be human-readable. (Requirement 4.4.3.5-1.3)

The Cast Vote Records (retrieved ballot images) shall not reveal the order in which they were voted. (Dangling ref: PrivacyShuffleCVR)

Section 6.4.1.4.3 Marksense and Punchcard

Test 6 Overvoting Simple Case

References: Requirement 4.4.3.3-7, Requirement 4.4.3.3-7.1

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Overvoting Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Three ballots shall vote for Ballot Position 1

Two ballots shall vote for Ballot Position 1 and Ballot Position 2

Two ballots shall vote for Ballot Position 2

Three ballots shall vote for Ballot Position 2 and Ballot Position 3

One ballot shall vote for Ballot Position 3

Four ballots shall vote for Ballot Position 1 and Ballot Position 3

One ballot shall vote for all three ballot positions

One ballot shall vote for none (undervote).

In addition to generating the usual reports, the VSTL shall perform four ad-hoc

queries to determine the number of overvotes combining ballot positions in each of the four applicable combinations (1+2, 1+3, 2+3, 1+2+3). (Requirement 4.4.3.3-7.1)

Section 6.4.1.4.4 Closed primaries

Test 7 Closed Primary Simple Case

References: Requirement 4.4.2-2.4, Requirement 4.4.3.2-1.2, Requirement 4.4.3.3-3.2, Requirement 4.4.3.3-4.1, Test 8, Test 8

Ballot format: Whig

2 1-of-M contests where M = 2.

The first contest shall be described as follows:

> This is the first contest in the Whig ballot format of the Closed Primary Simple Case Test. There are two candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Whig 1
> Whig 2

The second contest shall be described as follows:

> This is the second contest, a non-partisan office. There are two candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Whig 3
> Tory 3

Ballot format: Tory

2 1-of-M contests where M = 2.

The first contest shall be described as follows:

> This is the first contest in the Tory ballot format of the Closed Primary Simple Case Test. There are two candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

Tory 1
Tory 2

The second contest (non-partisan) shall be identical to the second contest in the Whig ballot format.

Reporting contexts: Single context.

Scenario:

Two Whig ballots shall vote for Whig 1 and Whig 3

One Whig ballot shall vote for Whig 2 and Tory 3

One Tory ballot shall vote for Tory 1 and Tory 3

Two Tory ballots shall vote for Tory 2 and skip the second contest (undervotes).

Additional pass criteria:

Separate counts for each party shall be reported in accordance with Requirement 4.4.3.3-3.2 and Requirement 4.4.3.3-4.1.

Section 6.4.1.4.5 Open primaries

Test 8 Open Primary Simple Case

References: Requirement 4.4.2-1.2.1, Requirement 4.4.2-2.5, Requirement 4.4.3.2-1.2, Requirement 4.4.3.3-3.2, Requirement 4.4.3.3-4.1

Ballot formats:

Ballot formats shall be identical to those in Test 7, except changing the name of the test case in the contest descriptions.

Reporting contexts: Single context.

Scenario: same as Test 7.

Additional pass criteria:

Separate counts for each party shall be reported in accordance with Requirement 4.4.3.3-3.2 and Requirement 4.4.3.3-4.1.

In a DRE system, the voter should be allowed to choose a party affiliation at the time of voting and vote the appropriate ballot format in privacy in accordance with Requirement 4.4.2-1.2.1.

Section 6.4.1.4.6 Write-ins

Test 9 Write-ins Simple Case

References: Requirement 4.4.2-2.6, Requirement 4.4.3.2-1.3

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Write-ins Simple Case Test. There are no candidates on the ballot. Write in at most one.

The only ballot position in the contest shall be a write-in opportunity.

Reporting contexts: Single context.

Scenario:

Four ballots shall write in First Write-In Candidate.

Three ballots shall vote for none (undervote).

Two ballots shall write in Second Write-In Candidate.

Section 6.4.1.4.7 Ballot rotation

Test 10 Ballot Rotation Simple Case

References: Requirement 4.4.2-2.7, Requirement 4.4.2-2.7.1, Requirement 4.4.3.2-1.4

Ballot format:

1 1-of-M contest where M = 3, with ballot rotation enabled.

The contest shall be described as follows:

This is the only contest in the Ballot Rotation Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

Candidate 1
Candidate 2
Candidate 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote for Candidate 1

Three ballots shall vote for Candidate 2

Two ballots shall vote for Candidate 3

Additional pass criteria:

In a DRE system, each candidate shall appear in each position on the ballot exactly three times in accordance with Requirement 4.4.2-2.7.1.

Section 6.4.1.4.8 Straight party voting

Test 11 Straight Party Voting Simple Case

References: Requirement 4.4.2-2.8, Requirement 4.4.3.2-1.5, Requirement 4.4.3.2-1.5.1, Requirement 4.4.3.2-1.5.2

Ballot format:

2 1-of-M contests.

The first contest shall be described as follows:

STRAIGHT PARTY. If you desire to vote a straight party ticket for all offices, vote for at most one party here. Votes for individual candidates in subsequent contests will override the straight party vote in those contests only.

The ballot positions shall be the following:

Whig

Tory

The second contest shall be described as follows:

This is the only contest in the Straight Party Voting Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

Ballot Position 1 (Whig)
Ballot Position 2 (Tory)
Ballot Position 3 (Independent)

Reporting contexts: Single context.

Scenario:

Two ballots shall vote straight party Whig and skip the second contest (allowing the straight party vote to be effective)

One ballot shall skip the straight party vote and vote for Ballot Position 1 (Whig)

One ballot shall votes straight party Tory but then vote for Ballot Position 1 (Whig)

Two ballots shall vote straight party Tory and skip the second contest

One ballot shall skip the straight party vote and vote for Ballot Position 2 (Tory)

One ballot shall vote straight party Tory but then vote for Ballot Position 3 (Independent).

(Expected result: Ballot Position 1 (Whig), 4; Ballot Position 2 (Tory), 3; Ballot Position 3 (Independent), 1.)

Section 6.4.1.4.9 Cross-party endorsement

Test 12 Cross-party Endorsement Simple Case

References: Requirement 4.4.2-2.8.1, Requirement 4.4.3.2-1.5.3

Ballot format:

2 1-of-M contests.

The first contest shall be described as follows:

STRAIGHT PARTY. If you desire to vote a straight party ticket for all offices, vote for at most one party here. Votes for individual candidates in subsequent contests will override the straight party vote in those contests only.

The ballot positions shall be the following:

Whig
Free-Soil
National
Federalist

The second contest shall be described as follows:

This is the only contest in the Straight Party Voting Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

Ballot Position 1 (Whig/National/Federalist)
Ballot Position 2 (Free-Soil)
Ballot Position 3 (Independent)

Reporting contexts: Single context.

Scenario:

One ballot shall vote straight party Whig and skip the second contest

Two ballots shall vote straight party Free-Soil and skip the second contest

Three ballots shall vote straight party National and skip the second contest

Two ballots shall vote straight party Federalist and skip the second contest

One ballot shall skip the straight party vote and vote for Ballot Position 2 (Free-Soil)

One ballot shall skip the straight party vote and vote for Ballot Position 3 (Independent).

Section 6.4.1.4.10 Split precincts

Test 13 Split Precinct Simple Case

References: Requirement 4.4.2-2.9, Requirement 4.4.3.2-1.6

Ballot format: District 1

1 1-of-M contest where M = 2.

The contest shall be described as follows:

> This is the only contest in the District 1 ballot format of the Split Precinct Simple Case Test. There are two candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> District 1 Candidate 1
> District 1 Candidate 2

Ballot format: District 2

1 1-of-M contest where M = 2.

The contest shall be described as follows:

> This is the only contest in the District 2 ballot format of the Split Precinct Simple Case Test. There are two candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> District 2 Candidate 1
> District 2 Candidate 2

Reporting contexts: Precinct 1, District 1, District 2, Jurisdiction. (Precinct 1 is split between District 1 and District 2.)

Scenario:

Three District 1 ballots shall vote for District 1 Candidate 1

Two District 1 ballots shall vote for District 1 Candidate 2

Six District 2 ballots shall vote for District 2 Candidate 1

Additional pass criteria:

Only the District 1 ballots shall be reported in the District 1 reporting context.

Only the District 2 ballots shall be reported in the District 2 reporting context.

All ballots shall be reported in the Precinct 1 and Jurisdiction reporting contexts.

Section 6.4.1.4.11 N of M voting

Test 14 N-of-M Simple Case

References: Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.7

Ballot format:

1 N-of-M contest where N = 2 and M = 3.

The contest shall be described as follows:

> This is the only contest in the N-of-M Simple Case Test. There are three candidates on the ballot. Vote for at most two.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote for Ballot Position 1 and Ballot Position 2

Three ballots shall vote for Ballot Position 1 and Ballot Position 3

Two ballots shall vote for Ballot Position 2 and nobody else (single undervote)

One ballot shall vote for none (double undervote).

Section 6.4.1.4.12 N of M voting + Write-ins

Test 15 N-of-M + Write-ins Simple Case

References: Requirement 4.4.2-2.6, Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.3, Requirement 4.4.3.2-1.7

Ballot format:

1 N-of-M contest where N = 2 and M = 3.

The contest shall be described as follows:

> This is the only contest in the N-of-M + Write-ins Simple Case Test. There are no candidates on the ballot. Write in at most two.

The ballot positions shall be two write-in opportunities.

Reporting contexts: Single context.

Scenario:

Four ballots shall write in "Write-in Candidate 1" and "Write-in Candidate 2"

Two ballots shall write in "Write-in Candidate 1" and "Write-in Candidate 3"

Four ballots shall write in "Write-in Candidate 2" and nobody else (single undervote)

One ballot shall vote for none (double undervote).

Section 6.4.1.4.13 Cumulative voting

Test 16 Cumulative Voting Simple Case

References: Requirement 4.4.2-2.11, Requirement 4.4.3.2-1.8

Ballot format:

1 cumulative voting contest where M = 3 and N(r) = 3.

The contest shall be described as follows:

> This is the only contest in the Cumulative Voting Simple Case Test. There are three candidates on the ballot. Cast at most three votes. You may cast multiple votes for the same candidate.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote once each for Ballot Position 1, Ballot Position 2, and Ballot Position 3

Three ballots shall vote twice for Ballot Position 2 and once for Ballot Position 3

Two ballots shall vote three times for Ballot Position 3

One ballot shall vote for none (undervote).

Section 6.4.1.4.14 Ranked order voting

Test 17 Ranked Order Voting Simple Case

References: Requirement 4.4.2-2.12, Requirement 4.4.3.2-1.9, Requirement 4.4.3.3-9

Ballot format:

1 1-of-M ranked order contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Ranked Order Voting Simple Case Test. There are three candidates on the ballot. Please rank them in order of preference.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots vote shall the ordering Ballot Position 1, Ballot Position 2, Ballot Position 3

Three ballots shall vote the ordering Ballot Position 2, Ballot Position 3, Ballot Position 1

Two ballots shall vote the ordering Ballot Position 3, Ballot Position 2, Ballot

Position 1.

Expected result:

Round 1:

- Ballot Position 1, 4
- Ballot Position 2, 3
- Ballot Position 3, 2

Round 2:

- Ballot Position 2, 5
- Ballot Position 1, 4

Section 6.4.1.4.15 Provisional / challenged ballots

Test 18 Provisional Ballots Simple Case

References: Requirement 4.4.2-2.13, Requirement 4.4.3.2-1.10, Requirement 4.4.3.3-1.3, Requirement 4.4.3.3-3.3, Requirement 4.4.3.3-4.2, Requirement 4.4.3.3-12

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Provisional Ballots Simple Case Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Two regular ballots shall vote for Ballot Position 1

Five provisional ballots shall vote for Ballot Position 1, and two shall be accepted

Three regular ballots shall vote for Ballot Position 2

One regular ballot shall vote for Ballot Position 3

Four provisional ballots shall vote for Ballot Position 3, and one shall be accepted

One provisional ballot shall vote for none (undervote), and shall be accepted.

Additional pass criteria:

The number of provisional / challenged ballots read and counted shall be reported in compliance with Requirement 4.4.3.3-3.3, and Requirement 4.4.3.3-4.2.

Section 6.4.1.4.16 Unofficial results generation

Test 19 Unofficial Results Simple Case

References: Requirement 4.4.3.3-16, Requirement 4.4.3.3-17

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the Unofficial Results Simple Case Test.
> There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

Four ballots shall vote for Ballot Position 1

Three ballots shall vote for Ballot Position 2

Two ballots shall vote for Ballot Position 3

One ballot shall vote for none (undervote).

In addition to the usual reports specified in the general test template, an unofficial vote data report shall be generated.

Additional pass criteria:

The unofficial report shall provide only aggregated results in unofficial reports, and not data from individual ballots. (Requirement 4.4.3.3-16)

The unofficial report shall clearly indicate that the results it contains are unofficial. (Requirement 4.4.3.3-17)

Section 6.4.1.5 Typical case tests

The purpose of typical case tests is to test the behavior of the voting system in scenarios that reflect typical use of the system in practice rather than artificial minimum and maximum conditions.

Election officials are encouraged to submit testing scenarios that more accurately reflect their use of voting systems in practice.

Section 6.4.1.5.1 Special instructions for Marksense and Punchcard

For systems claiming the *Marksense* or *Punchcard* profiles, all applicable typical case tests shall be executed at a tabulating rate no less than 30 ballots per minute,[17] or the maximum rate at which the tabulating equipment is documented to function reliably, whichever is less. To speed testing, a higher rate may be used if the vendor does not object.

Section 6.4.1.5.2 All systems

Test 20 1-of-M Typical Case

References: Requirement 4.4.1.1-2.1.1, Requirement 4.4.1.1-2.1.2, Requirement 4.4.2-2.3, Requirement 4.4.3.2-1.1

Assumptions:

$L_R \geq 10$

$L_C \geq 10$

$L_B \geq 75000$

$L_T \geq 38375$

Ballot format:

10 1-of-M contests where M = 10.

The contests shall be described as follows (substituting numbers from 1 to 10 for *r*):

> This is Contest *r* in the 1-of-M Typical Case Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to 10 for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Precinct 1, Precinct 2, ... through Precinct 100, Jurisdiction.

Scenario:

A total of 75000 ballots shall be cast. Precincts 1 through 50 shall each receive 750 - *n* ballots, for precinct number *n*. Precincts 51 through 100 shall each receive 700 + *n* ballots.

In all precincts,

> 345 ballots shall vote for ballot position 1 in every contest
> 345 ballots shall vote for ballot position 2 in every contest
> 1 ballot shall vote for ballot position 3 in contest 3 and undervote the rest
> 1 ballot shall vote for ballot position 4 in contest 4 and undervote the rest
> 1 ballot shall vote for ballot position 5 in contest 5 and undervote the rest
> 1 ballot shall vote for ballot position 6 in contest 6 and undervote the rest
> 1 ballot shall vote for ballot position 7 in contest 7 and undervote the rest
> 1 ballot shall vote for ballot position 8 in contest 8 and undervote the rest
> 1 ballot shall vote for ballot position 9 in contest 9 and undervote the rest
> 1 ballot shall vote for ballot position 10 in contest 10 and undervote the rest

In precincts 1 through 50, all remaining ballots shall vote for ballot position 1 in the first contest and undervote the rest.

In precincts 51 through 100, all remaining ballots shall vote for ballot position 2 in the first contest and undervote the rest.

Discussion:

This test is based loosely on the minimum acceptance test guidelines in Appendix J of [2]. It has been modified to remove the explicit requirement for a large number of

voting machines for testing - such are unlikely to be available for a system that is not yet qualified. The requirement for N-of-M voting has been removed to permit the test to apply to all systems.

Section 6.4.1.5.3 Marksense and Punchcard

Test 21 1-of-M Paper Typical Case

References: Requirement 4.4.3.3-7

Assumptions:

$L_R \geq 10$

$L_C \geq 10$

$L_B \geq 75000$

$L_T \geq 34500$

Ballot format:

10 1-of-M contests where M = 10.

The contests shall be described as follows (substituting numbers from 1 to 10 for *r*):

> This is Contest *r* in the 1-of-M Paper Typical Case Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to 10 for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Precinct 1, Precinct 2, ... through Precinct 100, Jurisdiction.

Scenario:

A total of 75000 ballots shall be cast. Precincts 1 through 50 shall each receive 750 - *n* ballots, for precinct number *n*. Precincts 51 through 100 shall each receive 700 + *n* ballots.

In all precincts,

345 ballots shall vote for ballot position 1 in every contest
345 ballots shall vote for ballot position 2 in every contest
1 ballot shall vote for ballot position 3 in contest 3 and undervote the rest
1 ballot shall vote for ballot position 4 in contest 4 and undervote the rest
1 ballot shall vote for ballot position 5 in contest 5 and undervote the rest
1 ballot shall vote for ballot position 6 in contest 6 and undervote the rest
1 ballot shall vote for ballot position 7 in contest 7 and undervote the rest
1 ballot shall vote for ballot position 8 in contest 8 and undervote the rest
1 ballot shall vote for ballot position 9 in contest 9 and undervote the rest
1 ballot shall vote for ballot position 10 in contest 10 and undervote the rest

In precincts 1 through 50, all remaining ballots shall overvote the first contest by voting for both ballot positions 1 and 3 and undervote the rest.

In precincts 51 through 100, all remaining ballots shall overvote the first contest by voting for both ballot positions 2 and 3 and undervote the rest.

Section 6.4.1.5.4 N of M voting

Test 22 N-of-M Typical Case

References: Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.7

Assumptions:

$L_R \geq 10$

$L_C \geq 10$

$L_B \geq 75000$

$L_T \geq 75000$

Ballot format:

There shall be 10 contests. The first contest shall be an N-of-M contest where M = N = 10.

The first contest shall be described as follows:

This is Contest 1 in the N-of-M Typical Case Test. Vote for at most 10.

The other 9 contests shall be 1-of-M contests where M = 10. These contests shall be described as follows (substituting numbers from 2 to 10 for *r*):

This is Contest *r* in the N-of-M Typical Case Test. Vote for at most one.

The ballot positions in all 10 contests shall be of the following form (substituting numbers from 1 to 10 for *c*):

Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Precinct 1, Precinct 2, ... through Precinct 100, Jurisdiction.

Scenario:

A total of 75000 ballots shall be cast. Precincts 1 through 50 shall each receive 750 - *n* ballots, for precinct number *n*. Precincts 51 through 100 shall each receive 700 + *n* ballots.

In precincts 1 through 50, all ballots shall vote for all 10 candidates in Contest 1.

In precincts 51 through 100, all ballots shall vote for only the first 8 ballot positions in Contest 1 (yielding two undervotes each).

In all precincts, for the remaining 9 contests,

345 ballots shall vote for ballot position 1 in every contest
345 ballots shall vote for ballot position 2 in every contest
1 ballot shall vote for ballot position 3 in contest 3 and undervote the rest
1 ballot shall vote for ballot position 4 in contest 4 and undervote the rest
1 ballot shall vote for ballot position 5 in contest 5 and undervote the rest
1 ballot shall vote for ballot position 6 in contest 6 and undervote the rest
1 ballot shall vote for ballot position 7 in contest 7 and undervote the rest
1 ballot shall vote for ballot position 8 in contest 8 and undervote the rest
1 ballot shall vote for ballot position 9 in contest 9 and undervote the rest
1 ballot shall vote for ballot position 10 in contest 10 and undervote the rest

In precincts 1 through 50, all remaining ballots shall vote for ballot position 1 in the first contest and undervote the rest.

In precincts 51 through 100, all remaining ballots shall vote for ballot position 2 in the first contest and undervote the rest.

Section 6.4.1.5.5 Discussion

Write-ins, etc.: need a typical case test that combines all typical profiles, if there is a most common combination.

Section 6.4.1.6 Capacity tests (covers testing of maximum ballot counting rate)

Following subsections are organized by compliance profiles. Functional tests are applicable only if the implementation statement claims the profile indicated in the subsection name.

Section 6.4.1.6.1 Special instructions for Marksense and Punchcard

For systems claiming the *Marksense* or *Punchcard* profiles, all applicable capacity tests shall be executed at the maximum speed or rate at which the tabulating equipment is documented to function reliably.

Section 6.4.1.6.2 All systems

Test 23 1-of-M Contest/Ballot Capacity

References: Requirement 4.4.1.1-2.1.1, Requirement 4.4.1.1-2.1.2, Requirement 4.4.2-2.3, Requirement 4.4.3.2-1.1

Ballot format:

L_R 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the 1-of-M Contest/Ballot Capacity Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

A total of L_B ballots shall be cast.

$\min\left(L_T, \max\left(\left\lceil \frac{L_B}{L_C} \right\rceil - c, 0\right)\right)$ ballots shall vote for Ballot Position *c* in every contest (*c* > 0).

Any ballots left over shall be blank (undervotes).

Test 24 Ballot format Capacity

References: Requirement 4.4.1.2.1-1

Assumptions:

$$L_T \geq \min(L_B, L_F) - L_C + 1$$

Ballot formats:

L_F ballot formats shall be constructed. These forms shall share the same set of contests. (They shall be identical except for their form identifications.)

There shall be L_R 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the Ballot Format Capacity Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in any of the ballot formats.

Reporting contexts: Single context.

Scenario:

$\min(L_B, L_F)$ ballots shall be cast.

The *n*th ballot shall use ballot format *n* and shall vote for ballot position $\min(n, L_C)$ in every contest.

Test 25 Vote Register Capacity

References: Requirement 4.4.3.3-1.2, Requirement 4.4.3.3-6

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Vote Register Capacity Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

> Unopposed Candidate

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. All shall vote for Unopposed Candidate.

Test 26 Undervote Register Capacity

References: Requirement 4.4.3.3-8

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Undervote Register Capacity Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

> Unopposed Candidate

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. All shall be blank.

Test 27 1-of-M Multi Capacity

References: Requirement 4.4.3.3-8

Assumptions:

$L_R \geq L_F$

Ballot formats:

L_F ballot formats shall be constructed. The first contest on each form shall be unique to that form. It shall be described as follows (substituting numbers from 1 to L_F for *f*):

> This contest appears only in Ballot Format *f*. Vote for at most one.

The ballot positions in these contests shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Form *f* Position *c*

Each ballot format shall contain $L_F - L_R$ other contests that are shared by all ballot formats. These contests shall be described as follows (substituting numbers from 1 to $L_F - L_R$ for *r*):

> This is Shared Contest *r* in the 1-of-M Multi Capacity Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in any of the ballot formats.

Reporting contexts: Single context.

Scenario:

For each ballot format *f* , for *f* from 1 to L_F, $\left\lfloor \frac{L_B}{L_F} \right\rfloor$ ballots shall be cast. Let $n = \left\lfloor \frac{L_B}{L_F} \right\rfloor \times L_F$. (This may total fewer than L_B ballots.)

In the contest that is unique to each ballot format,

$\min\left(L_T, \max\left(\left\lceil \frac{\left\lfloor \frac{L_B}{L_F} \right\rfloor}{L_C} \right\rceil - c, 0\right)\right)$ ballots shall vote for Ballot Position *c* (*c* > 0). Any ballots left over shall undervote the unique contest.

In all other (shared) contests, $\min\left(L_T, \max\left(\left\lceil \frac{n}{L_C} \right\rceil - c, 0\right)\right)$ ballots shall vote for Ballot Position *c* in every contest (*c* > 0). Any ballots left over shall undervote all of the shared contests.

Section 6.4.1.6.3 Marksense and Punchcard

Test 28 Overvote Register Capacity

References: Requirement 4.4.3.3-7

Ballot format:

1 1-of-M contest where M = 2.

The contest shall be described as follows:

> This is the only contest in the Overvote Register Capacity Test. There are two candidates on the ballot. Vote for one.

The ballot positions in the contest shall be the following:

> Ballot Position 1
> Ballot Position 2

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. All shall vote for both ballot positions (overvote).

Section 6.4.1.6.4 Write-ins

Test 29 1-of-M Write-in Capacity 1

References: Requirement 4.4.2-2.6, Requirement 4.4.3.2-1.3

Ballot format:

L_R 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the 1-of-M Write-in Capacity 1 Test. Vote for at most one.

The ballot positions from 1 to L_C-1 in each contest shall be of the following form (substituting numbers from 1 to L_C-1 for *c*):

> Contest *r* Ballot Position *c*

The final ballot position in each contest shall be a write-in opportunity.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. All shall write in "Write-in Candidate *r*" in every contest, substituting contest numbers 1 to L_R for *r*.

Test 30 1-of-M Write-in Capacity 2

References: Requirement 4.4.2-2.6, Requirement 4.4.3.2-1.3

Ballot format:

L_R 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the 1-of-M Write-in Capacity 2 Test. There are no candidates on the ballot. Write in at most one.

The only ballot position in each contest shall be a write-in opportunity.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T, L_W)$ ballots shall be cast. All shall write in "Write-in Candidate *n*" in every contest, substituting ballot numbers 1 to $\min(L_B, L_T, L_W)$

for *n* (each ballot shall vote for a different write-in candidate).

Section 6.4.1.6.5 Straight party voting

Test 31 1-of-M Straight Party Capacity

References: Requirement 4.4.2-2.8, Requirement 4.4.3.2-1.5, Requirement 4.4.3.2-1.5.1, Requirement 4.4.3.2-1.5.2

Ballot format:

The first contest shall be described as follows:

> STRAIGHT PARTY. If you desire to vote a straight party ticket for all offices, vote for at most one party here. Votes for individual candidates in subsequent contests will override the straight party vote in those contests only.

The only ballot position shall be the following:

> Whig

There shall be L_R-1 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R-1 for *r*):

> This is Contest *r* in the 1-of-M Straight Party Capacity Test. Vote for at most one.

The first ballot position in each contest shall be of the following form:

> Contest *r* Whig Candidate (Whig)

The remaining ballot positions in each contest shall be of the following form (substituting numbers from 2 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. Each one shall vote straight party Whig in the first contest and skip the remaining contests (allowing the straight party vote to be effective in every contest).

Section 6.4.1.6.6 N of M voting

Test 32 N-of-M Capacity

References: Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.7

Ballot format:

L_R N-of-M contests where N = M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the N-of-M Capacity Test. Vote for at most L_C.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. All shall vote for every candidate in every contest.

Section 6.4.1.6.7 N of M voting + Write-ins

Test 33 N-of-M Write-ins Capacity 1

References: Requirement 4.4.2-2.6, Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.3, Requirement 4.4.3.2-1.7

Ballot format:

L_R N-of-M contests where $N = \left\lfloor \frac{L_C}{2} \right\rfloor$ and M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the N-of-M Write-ins Capacity 1 Test. Vote for at most N.

The first M-N ballot positions in each contest shall be of the following form (substituting numbers from 1 to M-N for *c*):

> Contest *r* Ballot Position *c*

The final N ballot positions shall be write-in opportunities.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. $\left\lfloor \frac{\min(L_B, L_T)}{2} \right\rfloor$ ballots shall vote for the first N candidates in each contest. The remaining ballots shall write in N candidates of the form "Contest *r* Write-in *n*," for *n* from 1 to N, in each contest.

Test 34 N-of-M Write-ins Capacity 2

References: Requirement 4.4.2-2.6, Requirement 4.4.2-2.10, Requirement 4.4.3.2-1.3, Requirement 4.4.3.2-1.7

Ballot format:

L_R N-of-M contests where N = M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the N-of-M Write-ins Capacity 2 Test. There are no candidates on the ballot. Write in at most L_C choices.

All L_C ballot positions in each contest shall be write-in opportunities.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_T)$ ballots shall be cast. In every contest, every ballot shall write in L_C choices of the form "Contest *r* Write-in *c*," substituting numbers 1 to L_C for *c*.

Section 6.4.1.6.8 Cumulative voting

Test 35 Cumulative Voting Capacity

References: Requirement 4.4.2-2.11, Requirement 4.4.3.2-1.8

Ballot format:

L_R cumulative voting contests where M = N(*r*) = L_C.

The contests shall be described as follows (substituting L_C and numbers from 1 to L_R for *r*):

> This is Contest *r* in the Cumulative Voting Capacity Test. Cast at most L_C votes. You may cast multiple votes for the same candidate.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

A total of $\min\left(L_B, \left\lceil \frac{L_T}{L_C} \right\rceil\right)$ ballots shall be cast.

$\min\left(L_B, \left\lfloor \frac{L_T}{L_C} \right\rfloor\right)$ ballots shall cast L_C votes for the first ballot position in every contest.

Any ballot left over shall cast $L_T - \left\lfloor \frac{L_T}{L_C} \right\rfloor \times L_C$ votes for the first ballot position in every contest and undervote the rest.

Section 6.4.1.6.9 Provisional / challenged ballots

Test 36 Provisional Ballot Capacity

References: Requirement 4.4.2-2.13, Requirement 4.4.3.2-1.10, Requirement 4.4.3.3-1.3, Requirement 4.4.3.3-3.3, Requirement 4.4.3.3-4.2, Requirement 4.4.3.3-12

Ballot format:

L_R 1-of-M contests where M = L_C.

The contests shall be described as follows (substituting numbers from 1 to L_R for *r*):

> This is Contest *r* in the Provisional Ballot Capacity Test. Vote for at most one.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to L_C for *c*):

> Contest *r* Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

A total of $\min(L_B, L_V)$ provisional ballots shall be cast and accepted for counting.

$\min\left(L_T, \max\left(\left\lceil \frac{\min(L_B, L_V)}{L_C} \right\rceil - c, 0\right)\right)$ ballots shall vote for Ballot Position *c* in every contest (*c* > 0).

Any ballots left over shall be blank (undervotes).

Additional pass criteria:

The number of provisional / challenged ballots read and counted shall be reported in compliance with Requirement 4.4.3.3-3.3, and Requirement 4.4.3.3-4.2.

Section 6.4.1.6.10 Discussion

In all systems there is the possibility of a bottleneck in transmitting precinct results to central.

Unfortunately there is no way to test this without having a large number of machines to work with -- an unreasonable expectation for a system that is not even qualified yet.

Many more capacity tests could be written for different combinations of profiles. There should be at least one torture test for the most common combination of profiles, if there is one.

Section 6.4.1.7 Error case tests

While most tests verify that the system does things that it is required to do, error case tests verify that the system does not do things that it is required not to do. As with all tests, passing an error case test does not conclusively show that the system is conforming, but failing an error case test conclusively shows that the system is non-conforming.

Section 6.4.1.7.1 All systems

Test 37 Vote Register Overflow

References: Requirement 4.3.1.1.2-3, Requirement 4.3.1.1.2-3.1

Assumptions:

$L_B > L_T$

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Vote Register Overflow Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

> Unopposed Candidate

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast L_T+1 ballots, all voting for Unopposed Candidate.

Additional pass criteria:

A DRE system shall not enable the L_T+1th ballot (Requirement 4.3.1.1.2-3.1). The tabulator in a Marksense or Punchcard system shall not accept the L_T+1th ballot (Requirement 4.3.1.1.2-3).

An audit log record shall exist for the counter reaching capacity event (Dangling ref: AuditLogCounterCapacityReq).

Discussion:

Overflow of the ballot counter is also implicitly tested by all of these register overflow tests.

Test 38 Undervote Register Overflow

References: Requirement 4.3.1.1.2-3, Requirement 4.3.1.1.2-3.1

Assumptions:

$L_B > L_T$

Ballot format:

1 1-of-M contest where M = 1.

The contest shall be described as follows:

> This is the only contest in the Undervote Register Overflow Test. There is only one candidate on the ballot.

The only ballot position in the contest shall be the following:

> Unopposed Candidate

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast L_T+1 ballots, all of them blank.

Additional pass criteria:

A DRE system shall not enable the L_T+1th ballot (Requirement 4.3.1.1.2-3.1). The tabulator in a Marksense or Punchcard system shall not accept the L_T+1th ballot (Requirement 4.3.1.1.2-3).

An audit log record shall exist for the counter reaching capacity event (Dangling ref: AuditLogCounterCapacityReq).

Section 6.4.1.7.2 Marksense and Punchcard

Test 39 Overvote Register Overflow

References: Requirement 4.3.1.1.2-3

Assumptions:

$L_B > L_T$

Ballot format:

1 1-of-M contest where M = 2.

The contest shall be described as follows:

> This is the only contest in the Overvote Register Overflow Test. There are two candidates on the ballot. Vote for one.

The ballot position in the contest shall be the following:

> Ballot Position 1
> Ballot Position 2

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast L_T+1 ballots. All shall vote for both ballot positions (overvote).

Additional pass criteria:

The tabulator shall not accept the L_T+1th ballot (Requirement 4.3.1.1.2-3).

An audit log record shall exist for the counter reaching capacity event (Dangling ref: AuditLogCounterCapacityReq).

Section 6.4.1.7.3 DRE

Test 40 DRE Overvoting

References: Requirement 4.4.2-2.2.2

Ballot format:

1 1-of-M contest where M = 3.

The contest shall be described as follows:

> This is the only contest in the DRE Overvoting Test. There are three candidates on the ballot. Vote for at most one.

The ballot positions shall be the following:

> Ballot Position 1
> Ballot Position 2
> Ballot Position 3

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast one ballot that votes for both Ballot Position 2 and Ballot Position 3.

Additional pass criteria:

The DRE shall prevent the VSTL from voting for more than one ballot position (Requirement 4.4.2-2.2.2).

Section 6.4.1.7.4 DRE + Write-ins

Test 41 DRE Write-ins Overvoting

References: Requirement 4.4.2-2.2.2

Ballot format:

1 1-of-M contest where M = 2.

The contest shall be described as follows:

This is the only contest in the DRE Write-ins Overvoting Test. There is one candidate on the ballot. Vote for at most one.

The first ballot position shall be the following:

Ballot Position 1

The second ballot position in the contest shall be a write-in opportunity.

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast one ballot that both votes for Ballot Position 1 and writes in "Write-in candidate."

Additional pass criteria:

The DRE shall prevent the VSTL from voting for more than one ballot position (Requirement 4.4.2-2.2.2).

Section 6.4.1.7.5 N of M voting

Test 42 N-of-M Vote Register Overflow

References: Requirement 4.3.1.1.2-3, Requirement 4.3.1.1.2-3.1

Assumptions:

$$L_C \geq 10$$

$$L_B > \left\lfloor \frac{L_T}{10} \right\rfloor$$

Ballot format:

1 N-of-M contest where N = M = 10.

The contest shall be described as follows (substituting numbers from 1 to L_R for *r*):

This is the only contest in the N-of-M Vote Register Overflow Test. Vote for at most 10.

The ballot positions in each contest shall be of the following form (substituting numbers from 1 to 10 for *c*):

Ballot Position *c*

There are no write-in ballot positions in this ballot format.

Reporting contexts: Single context.

Scenario:

The VSTL shall attempt to cast $\left\lfloor \frac{L_T}{10} \right\rfloor + 1$ ballots, all of which vote for all 10 ballot positions.

Additional pass criteria:

A DRE system shall not enable the $\left\lfloor \frac{L_T}{10} \right\rfloor + 1$ th ballot (Requirement 4.3.1.1.2-3.1).

The tabulator in a Marksense or Punchcard system shall not accept the $\left\lfloor \frac{L_T}{10} \right\rfloor + 1$ th ballot (Requirement 4.3.1.1.2-3).

An audit log record shall exist for the counter reaching capacity event (Dangling ref: AuditLogCounterCapacityReq).

Discussion:

A likely fault is for the system to only check that the count is less than L_T before enabling a ballot (should be less than L_T-9). This fault is masked in this test if L_T is a multiple of 10. We could test with other values, but it is possible to mask the fault in all tests by making L_T a product of those values.

Section 6.4.1.7.6 Discussion

Mechanical / lever systems appear to be out of scope of the VVSG. But if not -- what is supposed to happen in mechanical / lever systems when a register reaches its limit?

Other potential error case tests, not yet written:

Remote data delivery with intentional disruption of transmission (implementation-dependent).

Exception handling (requires creating an exception somehow - for testability, suggest adding a requirement for the capability to generate test exceptions. Could be of use in *in situ* L&A testing.)

Section 7 Bibliography

[1] 2002 Voting Systems Standards, available from http://www.eac.gov/election_resources/vss.html.

[2] 1990 Voting Systems Standards, official version no longer available. Unofficial PDF produced by Joseph Lorenzo Hall, UC Berkeley, 2004-07-15, available at http://www.calvoter.org/issues/votingtech/vtstandards.html.

[3] Voluntary Voting System Guidelines, draft, 2005-06, available from http://guidelines.kennesaw.edu/vvsg/intro.asp.

[4] UML 2.0 Superstructure Specification, 2004-10-08, http://doc.omg.org/ptc/2004-10-02.

[5] IEEE Draft Standard for the Evaluation of Voting Equipment, draft P1583/D5.3.2b, 2005-01-04. Available from http://grouper.ieee.org/groups/scc38/1583/private/ (password-protected).

[6] Fred R. Byers, Care and Handling of CDs and DVDs -- A Guide for Librarians and Archivists, National Institute of Standards and Technology Special Publication 500-252, 2003-10, available from http://www.itl.nist.gov/div895/carefordisc/index.html.

[7] Recommended Security Controls for Federal Information Systems, National Institute of Standards and Technology Special Publication 800-53, 2005-02, available from http://csrc.nist.gov/publications/nistpubs/.

[8] ISO 9706:1994, Information and documentation -- Paper for documents -- Requirements for permanence. Available from ISO, http://www.iso.org/.

[9] ES&S International, Comment on NIST Preliminary Report: Disposition of 2002 VSS Requirements, 2005-04-14, available at http://vote.nist.gov/ECPosStat.htm.

[10] Capability Maturity Model Integration, http://www.sei.cmu.edu/cmmi/.

[11] Philippe A. Martin, Petri Net Linear Form (PNLF), in "Using PIPE and Woflan," http://meganesia.int.gu.edu.au/~phmartin/workflow/PIPE/, 2005-07-22.

[12] New Shorter Oxford English Dictionary, Clarendon Press, Oxford, 1993.

Notes

[1] Commercial equipment and materials are identified in order to describe certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

[2] The understanding that a report is not mutable is important to system integrity. For example, a vote data report represents a "snapshot" summary of vote data as it appeared at a specific time. Subsequent alterations to that vote data shall not result in any changes to the report that was generated, but a new report (with a later timestamp) may be generated to summarize the revised data. Analogously, a test lab forwards a report to the EAC at a specific time. If a system is sent back for retesting, the result is a new report. The original report is never "lost" due to subsequent changes.

[3] Volume 2, Appendix B, Section B.5: "Any uncorrected deficiency that does not involve the loss or corruption of voting data shall not necessarily be cause for rejection."

[4] Formerly, it was "at most 1 from group 2," but this proved incompatible with security requirements.

[5] Portions of this section are derived from Section 5.6.2.2 of [5].

[6] This material is from an unapproved draft of a proposed IEEE Standard, P1583. As such, the material is subject to change in the final standard. Because this material is from an unapproved draft, the IEEE recommends that it not be utilized for any conformance/compliance purposes. It is used at your own risk.

[7] Portions of this section are derived from Sections 5.6.2.2 and 6.6.4.2 of [5].

[8] In mathematical jargon, the word *domain* would be more appropriate than *range* for input variables; however, "range checking" is the common programming jargon.

[9] Portions of this paragraph are derived from Section 6.6.4.2, Paragraph i of [5].

[10] The voting system is required to be capable of counting and reporting totals for all candidates that are written in by voters. In some states, write-in votes are not counted unless they exactly match one of a list of registered, accepted write-in candidates. Voting systems may support reporting options that meet the requirements of such states without disruption to the counting logic.

[11] The use of preconditions and postconditions as we have recommended first appeared in C. A. R. Hoare, "An Axiomatic Basis for Computer Programming," Communications of the ACM, v. 12, n. 10, October 1969, pp. 576-580, 583, with ideas derived from Robert W Floyd, "Assigning Meanings to Programs," in J. T. Schwartz, ed., Mathematical Aspects of Computer Science: Proceedings of Symposia in Applied Mathematics, v. 19, American Mathematical Society, 1967, pp. 19-32.

[12] Informality is permitted here to bridge the gap between a programming language with informal semantics and the formality that we require. The size limit on callable units in source code (Requirement 4.3.4.1.1.3.1-1.2) is intended to keep them small enough that preconditions and postconditions can be validated by inspection.

[13] Based on finding, definition 6, in [12].

[14] The VSTL should rely on authoritative information regarding the archivalness of various storage media. See also [6] and [8].

[15] Requirement 4.3.6-1.3 and Requirement 4.3.6-1.4 indicate acceptable designs.

[16] Portions of this paragraph are derived from Section 6.6.4.2, Paragraph i of [5].

[17] Default tabulating rate is from [2] J-3.