DOMINION VOTING™

Our customers come first.

2.07 - Democracy Suite System Test And Verification

Version: 4.19::110

April 6, 2015

Contents

VVSG Trace Listing

Chapter 1

Introduction

This document represents a Democracy Suite Test and Verification Specification. It covers variety of system testing and verification procedures, including a complete list of applicable test cases.

VVSG 2005 vol II §2.1.1.1.e

VVSG 2005 vol II §2.7

1.1 Purpose

The purpose of this document is to describe the plans, procedures and data used during software development and system integration to verify system logic correctness, data quality and security.

1.2 Document Use

This document is intended for use with the Democracy Suite® 4.14 platform.

[redacted]

[redacted]

1.4 Applicable TDP Requirement Sections

- VVSG 2005, Volume II, Version 1.0, Section 2.7 System Test and Verification Specification
- VVSG 2005, Volume II, Version 1.0, Section 2.7.1 Development test Specifications
- VVSG 2005, Volume II, Version 1.0, Section 2.7.2 - National Certification Test Specification

1.5 Design Responsibility

Dominion Voting is the design authority.

1.6 Design Authorities and Allowed Authors

Noted above.

1.7 Document Status

This is a working specification for discussion and analysis. Details are subject to change.

1.8 Patent Status

Certain system concepts, as well as many implementation and construction details are protected by a series of U.S. and foreign patents pending.

Chapter 2

System Test and Verification Specification

VVSG vol II §2.7

The development, execution and maintenance of system test and verification specifications are the responsibility of Dominion Voting's Quality Assurance (QA) department. Dominion Voting's QA department follows Standard Operating Procedures that define the department's work across the system development cycle, and are subsequently described.

Key procedures followed by the QA department.

TestLink (version 1.8.3) is the primary tool used by the QA team. This tool implements a set of policies and procedures that help structure the department's work. Each Dominion Voting product is represented as a test project in TestLink. A number of test cases are identified and written for each test project. Test cases are organized in appropriate test suites, which group test cases according to the functionalities they are intended to verify.

A number of test plans are also defined for each test project. The QA manager creates a new build in TestLink for every testing cycle. In other words, each new build issued by Dominion Voting is recorded and tested in TestLink. Depending on the nature of the release (build), the QA manager will determine what test plan needs to be executed (Smoke Test, Sanity Test, Regression Test, etc.). For example, if a release is an unofficial intermediate development release, QA will execute Smoke and/or Sanity Test Plans. However, if a release is official, QA must execute a full Regression Test Plan.

The QA manager (or a test lead) assigns test cases from a test plan to testers. Testers execute these test cases and record the outcomes as either Pass or Fail. If testers discover a severe bug in the process of testing, they must add a test case to the appropriate test suite. New test cases are executed in upcoming test cycles to recreate these defects. Once all test cases have been performed, the test manager determines the overall status of the release by examining the number and severity of defects recorded. Each bug found is logged in JIRA[1]. See www.atlassian.com as a defect and scheduled for correction by the development lead according to its impact and severity level. The defect is tested in the next testing cycle once it has been fixed.

Each testing cycle produces an audit trail listing the outcomes of testing. The QA manager has access to the Reports and Metrics section of TestLink, where he/she can use a number of methods to report on any product (test project) release. Test cases are continuously maintained and updated. A test case's version number monitors its evolution.

TestLink is used to enforce most of the testing verification processes and procedures. For more information on the concepts and logic of the processes and procedures involved, please see the TestLink manual.

Testing of EMS requires in-depth knowledge of the system and experience using and testing all of the products involved. Therefore, all testing efforts are subject to adjustments and modifications according to the manager's discretion.

[1]JIRA is an Enterprise-grade issue tracking system

Please note that the test cases included in this document are always subject to change due to constant maintenance and evolution of the products they are designed to test.

2.1 Development Test Specifications

Plans, procedures, and data used during software development and system integration testing are used to verify system logic correctness, data quality, and security. These tests are described in the following sections and are used by the Dominion Voting Test Department.

The following sections detail:

- Test identification and design including
  - Pre-test conditions
  - Test structure
  - Test sequence or progression
  - Test conditions
- Standard test procedures, including any assumptions or constraints
- Expected test results
- Criteria for evaluating test results

2.1.1 Test Identification and Design

Test cases are identified and written for each of the products (called 'test projects' in TestLink). Each test case can cover one or more functionalities and each functionality may require one or more test cases to be executed for verification. Each release/build is tested according to test plans that are designed to verify different aspects of the product(s) tested and/or their integration points.

Test plans are comprised from a number of different test cases selected from one or more projects. Test plans are dynamic and evolve together with the products and, consequently, test cases. The design of test plans and higher level of testing efforts to be executed are the responsibility of the QA manager. Generally, a high level test design identifies the products to be tested, environments, conditions and pre-requirements, integration points, test plans and lastly test cases. After that, test scheduling, execution and reporting is performed.

2.1.1.1 Test Structure

Each test case, the following elements are defined/added by a tester:

- Name
- Summary
- Test Suite (assigns)
- Prerequisites
- Resources Required
- Steps

- Expected Results
- Notes
- Keywords
- Attached Files

The following elements are automatically assigned to the test case by the system (i.e. TestLink):

- Unique number (ID)
- Version
- Date Created (date)
- Created By (user name)
- Last Modified (date)

Additional actions a tester can perform on a test case:

- Edit Test Case
- Move/Copy Test Case
- Create a new Version
- Deactivate this Version
- Add to test plan

Each test case belongs to the Test Suite. A test suite can have one or more test cases and one or more test suites.

Test cases are never deleted. Old or outdated test cases are moved into specific test suites created for this purpose. As a result, the test execution records are kept for all test cases ever executed regardless of their current status/version. All test cases can be found in the *2.07 Democracy Suite System Test and Verification Specification - Test Cases* document.

2.1.1.2 Test Sequence or Progression

Test sequence and progression is defined by a combination of elements. Each project in TestLink has a set of defined test cases. Depending on the release tested (official or intermediate/development release) the test cases are assigned to a test plan. The selection of test cases and the order in which they are added to the test plan determines the sequence and progression in which they are executed.

The sequence and progression is subject to change according to the manager's discretion.

2.1.1.3 Test Conditions

Each defined condition may be verified by a single test case or a set of test cases which would, most often, comprise a test plan. Therefore, all conditions are defined by individual test cases, sets of test cases as well as any prerequisites and/or notes that may be included in the test case(s).

2.1.2 Standard Test Procedures

The standard test procedures are as described in the preceding sections.

2.1.3 Special Purpose Test Procedures

There are no special purpose test procedures.

2.1.4 Test data

All tests can be performed on any election event definition (contest, candidates, ballots, voting locations) and the test descriptions have been generalized to eliminate limitations. All tests are performed using end-to-end simulations without any use of simulated data. All test data is real in that the described tests either create the data during the test, or use existing data from preceding tests (listed as pre-conditions). The evaluation of test data resulting from testing can be performed using any audit trail (results tape, ballot image with results, consolidated results).

2.1.5 Expected Test Results

Expected test results are defined for each test case in a step-wise manner. Each test case consists of a number of steps that have to be executed. Each step has its associated expected result(s).

2.1.6 Criteria for Evaluating Test Results

Criteria for evaluating test results are defined in a hierarchical manner. First, each individual test case is marked as 'pass' or 'fail.' The outcome of the test case execution depends on whether the actual results of the test case match the predefined expected results for that test case. Test cases are executed as a part of the test plan(s) that the QA manager assigns to specific test analysts. Once all test cases are completed, the QA manager decides on the overall status of the testing effort. The decision on the overall status of the release/build is reached by analyzing the pass/fail ratio of all test cases executed as well as individual fail results and their resulting defect logs (bug number, impact and severity).

2.1.7 Verifying System Logic Correctness

System logic correctness is verified in a step-wise manner where for each step in the test process, the description of the test step is compared to the system behaviour/output and to the predefined expected results for each step.

2.1.8 Verifying System Data Quality

Data quality is only assessed in the test environment via comparison of expected and actual voting results. This examination can be performed by using any audit trail (results tape, ballot image with results, consolidated results).

2.1.9 Verifying System Security

System security is verified within various test procedures and pre-conditions. For example, by reviewing the Test Case “Open Project” in the RTR section of the appendix document, it is evident that the correct user name and password have to be provided in order to create a project. If correct credentials are provided, the user can successfully create a project. However, if the wrong credentials are provided the error message appears.

Security is verified throughout the testing efforts and covers all aspects of the system as specified in the VVSG 2005, vol I §7. In some cases, security verification will have a separate test suite for the particular project/product.

2.2 National Certification Test Specifications

Procedures for the verification and validation of overall software performance have been based on VVSG requirements and included in the test cases. These tests provide procedures for assessing and demonstrating the suitability of the software for election use. The basic concept is that single election simulations cannot individually demonstrate suitability of software for all possible applications. Multiple simulations are created which test bundles of functions in a matter that is similar to how different jurisdictions use different voting variations. Each simulation is created from scratch with sample information which may be past election data or dummy data.

2.2.1 Control and Data Input/Output

Pre-set data is not defined. A bundle of functions is combined into an election simulation (for example a General Election simulation) and the test is performed on a set of subdivisions and precincts. Data input is performed from scratch for each test. The manual data entry process is defined by the ‘Steps’ section listed for each test case and by the equivalence classes as defined in TDP *2.12 Democracy Suite Quality Assurance Program*, Section 2.6.1.2.

2.2.2 Acceptance Criteria

The functionalities under test are known before the start of the election, and are integrated into the design of the test. When the test is performed, these functions are examined and must meet the declared functionality in the Technical Data Package. For more details on release process see the TDP document titled *2.11 Democracy Suite Configuration Management Process*, Section *Software Baseline, Promotion and Release Process* and the TDP document titled *2.12 Democracy Suite Quality Assurance Program*, Section *Release Criteria*.

2.2.3 Processing Accuracy

The accuracy of every function tested must be 100%.

2.2.4 Data Quality Assessment and Maintenance

Data quality and accuracy is measured by determining the expected results of the simulation before any voting is performed. Test results are inspected to ensure they match the expected outcome for the specific input. The data is always within the boundaries of the election system, so any maintenance is inherent in the assessment of data quality. Data is not required to be exported at any time for inspection.

2.2.5 Ballot Interpretation Logic

Ballot interpretation logic is defined before the start of testing. The results of the voting and the response of the system to the voting variations are then known beforehand and listed in the Expected Results section of test cases.

2.2.6 Exception Handling

Exception handling can be validated and verified using some of the test cases specified in this document. The test cases are either grouped into exception handling test suits, or the exception handling is addressed in an individual test if the possible exception(s) are within the scope of that particular test. For an example of an exception handling test suite, see the document 2.07- System Test And Verification- Test Suites. For examples of error handling within an individual test case, see test cases in EED test cases section of the document 2.07- System Test And Verification- Test Suites.

2.2.7 Security

Security is an overall functionality that can be tested independently of voting variations. Sample test cases from this test document are used. These test cases are either grouped in relevant test suites, or the security aspects are tested within an individual test case. For more details please see 2.07- System Test And Verification- Test Suites.

2.2.8 Production of Audit Trails and Statistical Data

Statistical data is required for the determination of accuracy and data quality. In each election test, all election statistics are produced and examined. Audit trail production is an overall functionality that can be tested independently of voting variations. Sample test cases from this test document are used. In some cases the audit trail test cases are grouped in appropriate test suites or an individual test case spread across multiple test suites. For specific test cases please see the document 2.07- System Test And Verification- Test Suites.

2.2.9 Test Logs

As we mentioned in section 2.1.1.1, each test case belongs to the Test Suite. A test suite can have one or more test cases and one or more test suites. All test cases are saved in the TestLink. Example of Test Log, extracted from TestLink is presented as in following paragraph. The test log presents the author of the test log query, test case number, author of the test case,test case summary,

Smoke Test Test Report Project: ICP Author: [redacted] Printed by TestLink on 26/07/2011 2009 Testlink Community Table Of Contents

1 Test Suite : Smoke Test

Test Case**ICP-674: Set Date and Time at Startup**
Author:[redacted]

Summary:

To verify that the user is prompted to confirm the date and time at startup, and can set the date and time if desired.

Resources:

- Election files and iButton
- Console

Pretest conditions: ICP is off.

Steps:

1. Insert election cards into the ICP and power it on.
2. Insert the admin key and passcode when prompted.

3. Press NO to set the date and time.
4. Set the correct date and time values. Press DONE after each value is entered.
5. Press TABULATOR INFO. Verify in the console that ICP is displaying the correct date and time.

Test Case **ICP-108: Scanning a Ballot in All Orientations**

Author: [redacted]

Summary:

To ensure that a voter can scan and cast a ballot in all four orientations. Resources:

- Election files and iButton
- A valid ballot

Pre-test conditions:

ICP is on and at main screen (LCD displays "System Ready" message). Steps:

1. Scan and cast a ballot in all four orientations.

Expected Results:

1. ICP accepts the ballot in all four orientations. The ballot counter increases by one for each ballot cast.

Last Result: Passed
Build 4.5.2
Tester [redacted]
Testing notes Famous County
Keywords: P1
Integration Test
Sanity Test

Expected Results:

[redacted]

Last Result: Passed
Build 4.5.4
Tester [redacted]
Keywords: P1
Sanity Test