


# COMMONWEALTH OF PENNSYLVANIA DEPARTMENT'S OF PUBLIC WELFARE, INSURANCE, AND AGING

## INFORMATION TECHNOLOGY STANDARD

Name Of Standard: <b>Software Test Planning</b>	Number: <b>STD-EASS008</b>
Domain: <b>Application Domain</b>	Category: <b>Testing / Software Quality Assurance</b>
Date Issued: <b>05/04/2010</b>	Issued By Direction Of:
Date Revised: <b>02/25/2014</b>	 Shirley A. Monroe, Dir of Div of Technical Engineering

### Abstract:

The delivery of quality software is the result of thorough test planning and proper execution. The Software Test Plan acts as a road-map for all software testing within a project. It describes which phases of testing will be implemented during the various stages of software development. The Software Test Plan shall reflect the overall test strategy which has been drawn up for the testing function and should be written by the Test Manager for the project to which it pertains.

The Software Test Plan framework is based on the Software Development Methodology (SDM) planned for individual projects (i.e., Modified Water Fall, Rapid Application Development methods). While the SDM test plan framework outlines a consistent method to plan and execute testing of information systems, each test plan is unique and requires careful and comprehensive planning. A Software Test Plan should describe the overall planning efforts and test approach for all testing throughout a project. The Software Test Plan is used to define, align, and outline: a) test strategies and approaches, b) governance, c) resource requirements (i.e., people, places, and things), d) test coverage, e) test cycles and durations, f) ownership, g) pass/fail criteria, h) test schedule, i) business and technical alignment, and j) reporting. The test plan shall be a specific SDLC deliverable integrated into the project management and quality management plans for information systems projects.

The objective of test planning is: 1) Map out a comprehensive actionable test plan that ensures test coverage and test effectiveness, 2) Improve the integrity of

the IT solution development and delivery processes, and 3) Validate operational readiness and end product quality. Test coverage refers to a coverage metric that evaluates thoroughness of testing. Hence, the test design and parameters must thoroughly test and exercise all the systems business and technical functionality, features, components, interfaces, and capabilities linked back to specific business and/or systems requirement.

Test effectiveness refers to the fault-detecting ability of the software testing techniques based on the coverage specified and the all-use case and associated data set testing criterion. Although exhaustive testing is not always feasible, the intent is to validate the business solution or proposed technology's operational readiness, integrity, and reliability while eliminating defects from migrating into a live production environments through comprehensive software application testing, quality assurance, and quality control project management processes and procedures.

## **General:**

Software releases that involve major modifications or enhancements that change systems functionality and business operations (i.e., processes and procedures) are required to have a formal Software Release Test Document (SRTD) with integrated test plans. The Software Release Test Document has three core components: 1) Systems Test Plan (i.e., Unit, Module, System's Integration testing), 2) Performance and Compliance Test Plan (i.e., Security Vulnerability, Load & Performance, and ADA testing), and 3) User Acceptance Test Plan. Each component of the SRDT should be all inclusive documenting all software test phase plans, schedule, and results. Hence, the SRDT is a dynamic document in which tests phases are planned, established, approved, and executed throughout the project SDLC phases. Within the Project Management Framework, the Software Release Test Document and associated Software Test Plans components are created and executed during the appropriate SDLC phases throughout the project life cycle. The systems test plan should outline the test specifications, resources, and schedule associated with the solution build process (i.e., unit, module, and systems integration test phases) and established prior to beginning of development phase. Similarly, the user acceptance test plan should outline the test specifications, resources, and schedule associated with the systems acceptance test (SAT) phase and be established prior to systems integration testing phase completion. The Performance & Compliance Test Plan (i.e., Security Vulnerability, Load & Performance, and ADA testing) should outline the test specifications, resources, and schedule is established prior to beginning of implementation phase. The Software Release Test Document and associated components serve as actual project deliverables as well as a software system's life cycle management document of record. The individual Software Test Plans should address but is not limited to, the following elements:

- Identify and describe the test phases to be executed during the SDLC and their respective ownership (i.e., roles and responsibilities)
- Describe the methodology for each test phase including testing types to be executed and the elements of testing to be used to determine correctness of the system (i.e., functions, features, performance) or system component
- Identify the general test scope, objectives, test types, test scenarios, data-sets, environment, and logistics
- Identify the inherent interdependencies of executing and validating complex integrated functional tests, systems components, and interfaces
- Identify the entry, suspension, and exit criteria for each test and overall test phase
- Define test cycles and durations with mappings to testing resources and schedule allowing time for break-fix-retest iterations (i.e., systems regression testing and revalidation cycle times)
- Define test coverage specifications by aligning test ID, requirement ID, associated software systems modules/components, and corresponding functionality or attributes.
- Define expected outcomes and specific test pass/fail criteria
- Define testing staff and their required knowledge and skill-sets to execute tests, identify and document anomalies, and accurately assess operational readiness
- Outline governance for evaluating, confirming, and documenting anomalies and defects
- Define defect limits, thresholds and tolerances for rework and/or mitigation
- Define systems acceptance (Go or No-Go) criteria based on defect types and severity as well as evaluation of test results relative to overall systems functionality, integrity, performance, and capability to successfully support business operations.
- Define the schedule of test activities, deliverables, and milestones
- Define the monitoring activities and milestones required to evaluate actual progress to plan
- Define the reports that will be produced to communicate the progress of test execution and test results
- Manage test preparation and execution
- Test Plans and documents should be linked to a specific software version or product release cycle

Software Test Plans must be developed in coordination with and be accessible by appropriate project team and stakeholder entities. In addition, all schedule and work plan activities and roles and responsibilities required for the execution of the Software Test Plans must be integrated into the Project Management Plan and DPW defect management processes and procedures. All information in the Software Release Test Document and associated Software Test Plans must be consistent with the Project Management Plan and the related project documents of record.

The Software Release Test Document and associated Software Test Plans must be updated to maintain consistency with other project documents throughout the project life cycle and then be accessible in the future to designated staff for DPW software life cycle management, software quality assurance, and ITIL continuous improvement initiatives.

The test plan outline shall include but is not limited to the following components:

### **1. Software System Overview**

Describe the purpose of the system and the software to which this document applies. Describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

### **2. Scope of Testing**

Document all tests that will be conducted throughout each phase of the project. Any testing that will not be performed in each phase should also be identified (i.e., tests included and tests excluded). Scope definitions should include test coverage relative to functionality and /or systems components that will be tested and validated.

Identify the test level (e.g., unit test, software integration test, functional validation test, enterprise performance test) and contain a full identification of the system and the software to which this document applies, including as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

### **3. Test Objectives**

Provide the test objectives to be accomplished at this test level. The objectives determine the particular attributes of the application on which testing will focus, and will drive the selection of tests to be performed. They may be repeated, or modified, as necessary. Examples of test objectives include module logic, intra-system module interfaces, external interfaces, capacity and performance, stress testing, user interfaces, functional requirements, security, and end-to-end business scenarios.

### **4. Participating Organizations**

Identify the organizations, commonwealth agencies, and business partners that will participate in the testing at the test site(s) and the roles and responsibilities, including verification of memorandums of understanding (MOUs) for each organization.

### **5. Personnel (Business and Technical)**

Identify the number, type, and skill level of personnel needed during the test period at the test site(s), the dates and times they will be needed and any special needs, such as makeshift operation and retention of key skills to ensure continuity and consistency in extensive test programs.

## **6. Test Plan Approach**

This is a high level description of the approach to be used for the SRDT test plans to test the software for each test phase. This is a general description of the types of testing which will be employed in the SDLC phases and outlining key strategies, identifying requirements and potential constraints. Items to include:

- Requirements traceability
- Scope of test cases (degree of coverage)
- Test objectives
- Test cases will be creation, review, and approvals
- The use of an automated test tool vs. manual testing
- Who will be executing the tests (e.g. testers, end users, developers)
- Test preparation and configuration requirements
- Describe any security or privacy protection considerations.
- 

## **7. Risks and Assumptions**

Identify and describe possible risks and assumptions that may impact the testing. Examples of these include: schedule slippage and/or unavailability of needed software packages and databases.

## **8. Test Design Components**

### **Design Validation**

Ensure tests are structured to validate business and technical requirements.

### **Data Validation**

What types of data will require conversion and validation? What parts of the feature or function will use what types of data? What are the data types that test cases will address? Etc.

### **API Testing**

What level of API testing will be performed? What is justification for taking this approach (only if none is being taken)?

### **Content Testing**

Is your area/feature/product content based? What is the nature of the content? What strategies will be employed in your feature/area to address content related issues?

### **Systems-Resource Testing**

What systems resources does your feature use? Which are used most, and are most likely to cause problems? What tools/methods will be used in testing to cover low resource (memory, disk, etc.) issues?

## **Setup Testing**

What are the necessary requirements for a successful setup to test features, functionality, or systems components? What is the testing approach that will be employed to confirm valid setup of these elements?

## **Modes and Runtime Options**

What are the different run time modes the program can be in? Are there views that can be turned off and on? Controls that toggle visibility states? Are there options a user can set which will affect the run of the program? List here the different run time states and options the program has available. It may be worthwhile to indicate here which ones demonstrate a need for more testing focus.

## **Interoperability**

How will this software or product interact with other products? What level of knowledge does it need to have about other programs, program interaction and fundamental system changes? What methods will be used to verify these capabilities?

## **Integration Testing**

Go through each area in the software or product and determine how it might interact with other internal modules, components or external application software, shared services, and/or associated platform technologies. The test types and cases created should exercise and validate all integration points.

## **Systems Compatibility & Configurations (Clients & Servers)**

Is there a standard protocols and/or specific configurations requirements for the servers or that the clients are expected to use during the SDLC testing phases? How many and which clients are expected to use your feature? Are there subtleties in the interpretation of standard protocols that might cause incompatibilities at the desktop or server level? How will the testing approach validate client and/or server compatibility? Is your server suited to handle ill-behaved clients? Are there non-standard, but widely practiced use of your protocols that might cause incompatibilities?

## **Environment/System - General**

Are there issues regarding the environment, system, or platform that should get special attention in the test plan? What are the run time modes and options in the environment that may cause difference in the feature? Are there platform or system specific compliance issues that must be maintained or reconfigured to support production environments?

## **Configuration**

Are there configuration issues or requirements regarding hardware and software in the environments that may be used in the test plan? Are there any internal components and/or external third party product configurations or monitors required during the test durations?

## **User Interface**

List the items in the feature that explicitly require a user interface. Is the user interface designed such that a user will be able to use the feature satisfactorily? How will the interface testing be approached?

## **Performance & Capacity Testing**

How fast and how much can the feature do? Does it do enough fast enough? What testing methodology will be used to determine this information? What criterion will be used to indicate acceptable performance? If modifications of an

existing product, what are the current metrics? What are the expected major bottlenecks and performance problem areas on this feature?

### **Scalability**

Is the ability to scale and expand this feature a major requirement? What parts of the system or specific feature are most likely to have scalability problems? What approach will testing use to define the scalability issues in the feature?

### **Stress Testing**

How does the system of feature react when pushed beyond its performance and capacity limits? How is its recovery? What is its breakpoint? What is the user experience when this occurs? What is the expected behavior when the client reaches stress levels? What testing methodology will be used to determine this information? What area is expected to have the most stress related problems?

### **Volume Testing**

Volume testing differs from performance and stress testing in so much as it focuses on doing volumes of work in realistic operational or production environments. Run the software as expected user will - with certain other components running, or for so many hours, or with data sets of a certain size, or with certain expected number of repetitions that mirror day-to-day business environments as well as peak business cycles.

### **International Issues**

Confirm localized functionality, that strings are localized and that code pages are mapped properly. Assure program works properly on localized builds, and that international settings in the program and environment do not break functionality. How is localization and internationalization being done on this project? List those parts of the feature that are most likely to be affected by localization. State methodology used to verify International sufficiency and localization.

### **Robustness**

How stable is the code base? Does it break easily? Are there memory leaks? Are there portions of code prone to crash, save failure, or data corruption? How good is the program's recovery when these problems occur? How is the user affected when the program behaves incorrectly? What is the testing approach to find these problem areas? What is the overall robustness goal and criteria?

### **Error Testing**

How does the program handle error conditions? List the possible error conditions. What testing methodology will be used to evoke and determine proper behavior for error conditions? What feedback mechanism is being given to the user, and is it sufficient? What criteria will be used to define sufficient error recovery?

### **Usability**

What are the major usability issues on the feature? What is testing's approach to discover more problems? What sorts of usability tests and studies have been performed, or will be performed? What is the usability goal and criteria for this feature?

### **Accessibility**

Is the feature designed in compliance with accessibility guidelines? Could a user with special accessibility requirements still be able to utilize this feature? What is the criteria for acceptance on accessibility issues on this feature? What is the testing approach to discover problems and issues?

## **User Scenarios**

What real world user activities are you going to try to mimic? What classes of users (i.e. secretaries, artist, writers, animators, construction worker, airline pilot, shoemaker, etc.) are expected to use this program, and doing which activities? How will you attempt to mimic these key scenarios? Are there special niche markets that your product is aimed at (intentionally or unintentionally) where mimic real user scenarios is critical?

## **Boundaries and Limits**

Are there particular boundaries and limits inherent in the feature or areas that deserve special mention here? What is the testing methodology to discover problems handling these boundaries and limits?

## **Operational Issues**

If your program is being deployed in a data center, or as part of a customer's operational facility, then testing must, in the very least, mimic the user scenario of performing basic operational tasks with the software.

## **Data Set, Files Refresh and Backup**

Identify all data sets and files required to conduct the tests, and indicate how those will be secured, refreshed, and backed up for individual test cycles. Also identify key services that are turned off or remain running, determine whether or not it is possible to backup the data and still keep services or code running.

## **Recovery**

If the program goes down, or must be shut down, are there steps and procedures that will restore program state and get the program or service operational again? Are there holes in this process that may make a service or state deficient? Are there holes that could provide loss of data. Mimic as many states of loss of services that are likely to happen, and go through the process of successfully restoring the software application or service.

## **Archiving**

Based on the Information Life Cycle management requirements, a test should be constructed to validate such business and system requirements to ensure proper execution and retrieval. Is archival of data going to be considered a crucial operational issue on your feature? If so, is it possible to archive the data without taking the service down? Is the data, once archived, readily accessible?

## **Monitoring**

Does the service have adequate monitoring messages to indicate status, performance, or error conditions? When something goes wrong, are messages sufficient for operational staff to know what to do to restore proper functionality? Are the "heartbeat" counters that indicate whether or not the program or service is working? Attempt to mimic the scenario of an operational staff trying to keep a service up and running.

## **Migration**

Is there data, script, code or other artifacts from previous versions that will need to be migrated to a new version? Testing should create an example of installation with an old version, and migrate that example to the new version, moving all data and scripts into the new format. Need to identify all data files, formats, or code that would be affected by migration, the solution for migration, and how testing will approach each.



### **Special Code Profiling and Other Metrics**

How much focus will be placed on code coverage? What tools and methods will be used to measure the degree to which testing coverage is sufficiently addressing all of the code?

### **Compliance Certifications**

Identify the compliance certifications that are expected to be completed. Define test methods and technologies required to secure certifications. Compliance certifications may be required for user interfaces, performance, interoperability, interfaces, and security.

### **General Test Conditions**

Describe conditions that apply to all of the tests or to a group of tests. For example: “each test shall include nominal, maximum, and minimum values;” “each test of type x shall use live data;” “execution size and time shall be measured for each software item.” Include a statement of the extent of testing to be performed and rationale for the extent selected. Express the extent of testing as a percentage of some well-defined total quantity, such as the number of samples of discrete operating conditions or values, or other sampling approach. Provide the approach to be followed for retesting/regression testing.

### **Test Cycles, Durations & Progression**

Define the test cycles and durations to determine level of effort and time schedules. Factor in break-fix-retest iterations as well. In cases of progressive or cumulative tests, define the planned sequence or progression of tests.

### **Test Phase Transition Criteria**

The following describe required criteria in order for testing to move from one state to another.

### **Entry Criteria**

List all criteria that must be met in order for test execution to begin. Possible items to list include:

- Test plan approved
- Test environment stable and ready
- Test cases written and approved
- Test tools ready
- Previous test phase’s exit criteria met
- Test resources available

### **Exit Criteria**

List all criteria that must be met in order for this test phase to be considered complete. Possible items for inclusion are:

- Test case completion
- Number and severity of open defects
- Passing of test objectives

### **Suspension Criteria**

This section should include criteria or conditions that if they occur, testing should be stopped. This is to avoid instances when the test team is asked to continue testing in an attempt to meet published schedules when in reality the software is not ready for testing.

### **Resumption Criteria**

In this section list criteria that must be met before testing can resume, in the event testing is suspended.

## **9. Test Plan Logistics & Coordination**

### **Document Overview**

Describe the relationship of the Software Test Plan to related project management plans. Describe how this test plan fits into the overall document structure that exists. This may include:

- Overall test strategy
- Other test phase test plans
- Company or group/department-wide test documents (e.g. Software Quality Assurance Plan)
- List the number, title, revision, date, and source of all documents referenced in this plan.

### **Test Plan Approvals**

List everyone who has to either review or approve the test plan document in this table, along with their project role.

Name	Project Role	Approval Date	Responsibility
			Approver
			Reviewer

### **Software Test Site Environments**

Describe the software test environment at each intended test site. Reference may be made to the Software Development Plan (SDP) for resources that are described there.

### **Site Identification**

Name of test site(s). Identify one or more test sites to be used for the testing, describe the software test environment at each site. If multiple test sites use the same or similar software test environments, they may be discussed together. Referencing earlier descriptions may reduce duplicative information among test site descriptions.

### **Software**

Identify by name, number, and version, as applicable, the software (e.g., operating systems, compilers, communications software, related applications software, databases, input files, code auditors, dynamic path analyzers, test drivers, preprocessors, test data generators, test control software, other special test software, post-processors) necessary to perform the planned testing activities at the test site(s). Describe the purpose of the software, describe its media (tape, disk, etc.), identify software expected to be supplied by the site, and identify any classified processing or other security or privacy protection issues associated with software.

### **Hardware and Firmware**

Identify by name, number, and version, as applicable, the computer hardware, interfacing equipment, communications equipment, test data reduction equipment, apparatus such as extra peripherals (tape drives, printers, plotters), test message generators, test timing devices, test event records, etc., and firmware that will be used in the software test environment at the test site(s). Describe the purpose of the hardware or firmware, state the period of usage and the number of each needed, identify those that are expected to be supplied by the site, and identify any classified processing or other security or privacy protection issues associated with the hardware or firmware.

### **Other Materials**

Identify and describe any other materials needed for the testing at the test site(s). These materials may include manuals, software listings, media containing the software to be tested, media containing data to be used in the tests, sample listings of output, and other forms of instructions. Identify those items that are to be delivered to the site and those that are expected to be supplied by the site. Include the type, layout, and quantity of the materials, as applicable. Identify any classified processing or other security or privacy protection issues associated with the items.

### **Proprietary Nature**

Acquirer's Rights, and Licensing. Identify the proprietary nature, acquirer's rights, and licensing issues associated with each element of the software test environment.

### **Installation, Testing, and Control**

Describe the developer's plans for performing each of the following, possibly in conjunction with personnel at the test site(s): a. Acquiring or developing each element of the software test environment, b. Installing and testing each item of the software test environment prior to its use, c. Controlling and maintaining each item of the software test environment.

### **Resource Requirements and Commitments**

Identify the number of business and technical resources required and the domain knowledge and skill level of personnel needed during the test period.

Define the level of involvement, locations, and the dates and times they will be needed as well as any special needs or accommodations to ensure continuity and consistency in extensive test programs.

### **Data Recording, Reduction, and Analysis**

Identify and describe the data recording, reduction, and analysis procedures to be used during and after the tests identified in the individual test plans. Include, as applicable, manual, automatic, and semi-automatic techniques for recording test results, manipulating the raw results into a form suitable for evaluation, and retaining the results of data reduction and analysis.

### **Project-unique identifier of a test**

Identify a test by project-unique identifier and provide the information specified below for the test. Reference may be made as needed to the general information.

- Test objective – identify the specific test objective for this test
- Test scope – describe the depth and breadth of testing to be applied to this specific test
- Test type or class
- Test requirements – Identify the specific requirements from the BRD or SRD, to be validated by this test
- Special requirements (for example, 48 hours of continuous facility time, simulation, extent of test, use of a special input or database.)
- Type of data to be recorded
- Type of data recording/reduction/analysis to be employed.
- Assumptions and constraints, such as anticipated limitations on the test due to system or test conditions-timing, interfaces, equipment, personnel, database, etc.
- Safety, security, and privacy protection considerations associated with the test

### **Test Work Plan and Schedules.**

Provide or reference the schedules for conducting the tests identified in this plan. Include:

- A listing or chart depicting the sites at which the testing will be scheduled and the time frames during which the testing will be conducted.
- Describe the total scope of the planned testing. Describe each test to be performed in a separate subsection for each function or software item to be tested.
- Define items and systems to be tested. Identify a function, software item, subsystem, system, or other entity by name and project-unique identifier. Describe each test planned for the item(s). (Note: the “tests”

in this plan are collections of test cases. There is no intent to describe each test case in this document.)

- A schedule for each SRTD test plan should be developed and included in the master project plan outlining the tasks, resources, dependencies, timeline, and deliverables. .
- A schedule for each test site depicting the activities and events listed below, as applicable, in chronological order with supporting narrative as necessary.
- On-site test period and periods assigned to major portions of the testing
- Pretest on-site period needed for setting up the software test environment and other equipment, system debugging, orientation, and familiarization
- Collection of database/data file values, input values, and other operational data needed for the testing
- Conducting the tests, including planned retesting.
- Preparation, review, and approval of the Software Test Reports

**Orientation Plan**

Describe any orientation and training to be given before and during the testing. This training may include user instruction, operator instruction, maintenance and control group instruction, and orientation briefings to staff personnel. If extensive training is anticipated, a separate Training Plan may be developed and referenced here.

**10. Business Requirements Traceability & Test Coverage**

Traceability from each test identified to the applicable business requirements identification number as defined in the BRD. The test coverage is defined by forming a cross-reference map to requirements, functionality being tested, the test scenarios used to test and validate functionality, and the specific software modules that will be exercised for the particular Test ID. The criticality quantifies the level of importance of the test relative to the success or failure of the function’s impacts on overall operational readiness and Go/No-Go decision criteria. An example is shown in figure 1 below:

Test ID	Test Name	Function	Scenario	Module (s)	Requirements ID Numbers	Criticality	Pass/Fail Criteria

**Figure 1**

### 11. System Requirements Traceability

Traceability from each test identified to the applicable systems requirements identification number as defined in the SRD. The test coverage is defined by forming a cross-reference map to requirements; systems attribute being tested, the test scripts used to test and validate functionality, and the specific component that will be exercised for the particular Test ID. The criticality quantifies the level of importance of the test relative to the success or failure of the attribute's impact on overall operational readiness and Go/No-Go decision criteria. An example is shown in figure 2 below:

Test ID	Test Name	Attributes	Test Scripts	Component	Systems Requirements ID Numbers	Criticality	Pass/Fail Criteria

Figure 2

### 12. Test Case Tracking

Describe how the status of each test case will be tracked for each test iteration. Also include the criteria for deciding how to determine the status of each test case (e.g. what needs to happen for a test case to be passed, failed, blocked, etc...). If a test management tool exists, references can be made to the documented processes from the tool.

### 13. Test Defect Tracking

Tracking detail:

- What tool is utilized for defect tracking
- Statuses used and what they mean
- Classification definitions
- The defect workflow, including roles (i.e. who makes the determination that a defect is ready to test, or can be closed)

Defect management for defect discovery, documenting, validating, tracking, resolution should follow DPW standards and guidelines for each test plan.

### 14. Test Discrepancies

This section shall contain a description of the procedures that will be used to capture, configuration control, and clear software discrepancies identified during tests. These procedures will be consistent with DPW Configuration Management policy and project governance processes.

## 15. Appendices

### **APPENDIX A - ACRONYMS**

Describe the acronyms as they are used in the plan.

### **APPENDIX B - DEFINITIONS**

Describe the key terms as they are used in the plan.

### **APPENDIX C - REFERENCES**

Provide a complete list of documents referenced in the text of the plan. Each reference shall contain document number, title, revision number and date. References will include but is not limited the following items:

**Policy and Regulation:** Policies or laws that give rise to the need for this plan

**DPW Policy and Standards:** DPW policies and standards that give rise to the need for this plan

**Other Life Cycle Documents:** Other plans or task descriptions that elaborate details of this plan

### **ADDITIONAL APPENDICES**

Additional appendices may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each annex shall be referenced in the main body of the document where the data would normally have been provided. Appendices may be bound as separate documents for ease in handling or appended to the end of the document. Additional appendices shall be lettered alphabetically (D, E, etc.).

### **Software Test Planning Guideline References:**

This document contains best practices, suggestions, and ideas that were gathered from experience or study of the following sources.

- IEEE Standard for Software Test Documentation (ANSI/IEEE Standard 829-1983)
-

- Defense Finance and Accounting Service – Software Test Plan
- www. [CarnegieQuality.com](http://CarnegieQuality.com).

**Standard Revision Log:**

<b>Change Date</b>	<b>Version</b>	<b>Change Description</b>	<b>Author and Organization</b>
04/29/2010	1.0	Initial creation	Thomas King
12/28/2010	1.1	Reviewed Content – No Changes	Thomas King
02/25/2014	1.2	Reviewed Content – Updated DEA Division Director	Michael Light