

## Introduction

---

### ***Welcome to your Software Tester: Elite Pocket Testing Guide!***

This guide contains the most portable, every-day actions and information you will need as a tester. Once you have read Software Tester: Elite, this guide will serve as a ready-made reference throughout your testing day.

Use this pocket testing guide whenever you need to test. Or if you're not testing, but just starting out in the software industry, this guide can help keep you focused on the important keys to your development.

Keep this with you and reference it whenever you need a little reminder. Learn the steps to take and make them habits! Pretty soon, you won't need this pocket guide anymore.

### **Happy Bug Hunting!!**

~ 1 ~

[www.Successful-Quality-Assurance.com](http://www.Successful-Quality-Assurance.com)

***Introduction..... 1***  
***Master Elite Skills.....3***  
***Where to Test.....6***  
***You've Found a Bug...Now What?.....9***  
***It's Time for Bug Regression! ..... 10***  
***Pocket Glossary..... 11***  
***Status Report Template... ..... 18***  
***Bug Template..... 19***

## Master Elite Skills...

---

### **Communication**

- Written and Verbal
- Be clear and concise and give respect
- Check your spelling

### **Bulletproof Bug Writing**

- Double and triple check until “bulletproof” is a habit
- Check for dupes first
- KISS
- Unquestionable Reproducibility
- Detailed, unambiguous Expected Result (cite documentation whenever possible)
- Check your spelling
- Review before submitting

### **Managing Expectations**

- Control what you agree to
- Communicate your understanding
- Know format and content of expected results
- Deliver results or communicate delay ASAP

### **Attention to Detail**

- Document everything
- No nuance is too small
- Use your detailed notes to provide impactful answers

### **Asking the Right Questions**

- What if...?
- Does product meet intended need?
- Does product conflict with target market's other software?
- Could usability be more intuitive?
- Are user paths clear?
- Is product a benefit or burden? And why?

### **Be Solution-Oriented and Proactive**

- Review docs
- Highlight incomplete docs or design
- Draw focus to unanswered questions
- Be aware of your team and save them time
- Find opportunities to create solutions
- Create an environment of improvement

### **Master Your Bugbase**

- Know all P1s (at least)
  - Review bugs opened each day
  - Review bugs closed each day
- Master the search function
- Learn how to export

### **Deliver Stellar Reports**

- Highlight major issues at the top of reports
- Detail anything that impedes testing
- Summarize progress and status
- Update expected completion dates
- Use hard numbers data

### **Deliver on Your Word**

- Under-promise and Over-deliver
- Remember: Your professional reputation is always on the line – build it

### **Never Stop Learning**

- Thinking is the hardest work there is, which is probably the reason why so few engage in it. ~Henry Ford
- Employ your time in improving yourself by other men's writings so that you shall come easily by what others have labored hard for. ~ Socrates
- Start somewhere, anywhere...and keep going! You are worth it!

## Where to Test...

---

Use these as points of reference as you master the art of testing. Here are, historically, common places where weaknesses in software can be exploited. Happy bug hunting!

### **Data Capture**

- How many characters can you type into a field?
- How many characters can you Copy and **Paste** into a field?
- Which characters can you enter by typing?
- Which characters can you enter by Copy and Paste?
- Are stated rules being enforced?
- Is <Tab> order logical?

### **Selectors**

- Left and right mouse click very quickly on a selection
- Double-click a selection
- Quickly one and then another selection
- Click and drag prior to selection
- Select any blank line item option

### **ReadMe.txt**

R.T.F.M. = Read the Manual

- Is the ReadMe.txt accurate?
- Is it up to date (product version#'s, copyright, etc.)?
- Verify all contact info listed (email, phone #'s, etc.)
- Validate with grammar and spell checker

### **Transitions**

- One screen to next screen
- Level transition
- Scene transition
- User transition
- User rights transition

### **Interrupts**

Create an interrupt while the program is not "expecting" one:

- While it is loading
- While reading information
- While writing information

### **Borders**

If testing a GUI, check the edges of each selectable asset

- Is selectable area appropriate?
- Is area outside of asset bounding box selectable?

### **Defaults**

- Do default selections produce expected result?
- Install to NON-Default directory
  - Do shortcuts recognize this install
  - All saved information accurate

### **Un-Install**

Un-Install the program and see if it really does "Un-Install"

- Upon Un-Install, does the program report all uninstalled?
- Validate Un-Install user facing message's accuracy
- Search hard drive(s) for remnants
- Search registry for remnants

# You've Found a Bug...

## Now What?

---

1. Always **check for duplicates** first
  - i. Has it already been entered?
2. **Eliminate the Variables**
3. **Establish the Reproducibility**
4. **Define the Impact**
5. **Enter the bug**
  - i. **Brief Description:** A single, succinct sentence
  - ii. **Expanded Description:** Verbose explanation
  - iii. **Reproducibility:** Always, Intermittent (x-out-of-y), or Seen Once
  - iv. **Steps to Reproduce:** Each step a single action
  - v. **Result:** The bug, what just happened
  - vi. **Expected Result:** What should have happened (cite doc)
  - vii. **Severity:** Impact to the user
  - viii. **Priority:** Not for QA (unless otherwise specified)
  - ix. **Status:** Begins Open/Active
  - x. **Assignee:** Ensure proper bug assignment
6. **Re-read** and Review
7. **Spell check** and correct
8. **Review**
9. **Submit**

~ 9 ~

[www.Successful-Quality-Assurance.com](http://www.Successful-Quality-Assurance.com)

## It's Time for Bug Regression!

---

1. **Know the Version**
  - i. Make sure you know (and have) the version that contains the fix
2. **Understand the Fix**
  - i. Ensure you know what the fix is
3. **Validate Appropriately**
  - i. Consider the Reproducibility listed
  - ii. Test until you are confident
4. **Comment Thoroughly**
  - i. What you did
  - ii. What happened
  - iii. What action you are taking
  - iv. Be clear and don't skimp!
5. **Assign Accurately**
  - i. Make sure you assign the bug to the correct person based on the results of your regression testing

## Pocket Glossary

---

- **Acceptance Testing** – Testing conducted to determine whether a software build is of acceptable quality to test further. Normally performed to validate the software meets a set of agreed acceptance criteria.
- **Ad Hoc Testing** – A testing in which the tester tries to break the system by randomly trying the system's functionality.
- **Agile Testing** – Testing practice for projects using agile methodologies, treating development as the customer of testing and emphasizing a test-first design paradigm.
- **Alpha** – Alpha is the test phase in which the product is usable in a test environment but not necessarily bug-free
- **Automated Testing** – Testing employing software tools which execute tests without manual intervention
- **Benchmark Testing** – Tests that use representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration
- **Beta** – Beta is the test phase in which the product is feature complete. This version may be distributed for limited public testing and feedback

- **Beta Testing** – Testing a release of a product conducted by the public
- **Black Box Testing** – Testing based on an analysis of the specification of a piece of software without reference to its internal workings
- **Boundary Testing** – Testing of the program’s own rules, from within
- **Bug** – A fault in a program which causes the program to perform in an unintended or unanticipated manner. An anomaly, defect, error, exception, or fault
- **Bugbase** – The database in which all defects (bugs) are logged, prioritized, and tracked. (see Defect Tracking System)
- **Compatibility Testing** – Testing how well software performs in a particular hardware/software/operating system/network/etc. environment.
- **Defect** – See “Bug”
- **Defect Tracking System** – A tool used by a project team to organize, prioritize, and track all defects (bugs) related to the project. The engine most often used is a database, although projects may also use a spreadsheet
- **Endurance Testing** – Checks for memory leaks or other problems that may occur with prolonged execution

- **End-to-End Testing** – Testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.
- **Final Candidate** – A Release Candidate that meets the criteria to have the final tests performed before Gold Master is declared
- **Functional Specification** – A document that describes in detail the characteristics of the product with regard to its intended features.
- **Functional Testing** – Testing the features and operational behavior of a product to ensure they correspond to its specifications. (See Black Box Testing)
- **Gold Master** – A shippable version of the product
- **Gray Box Testing** – A combination of Black Box and White Box testing methodologies to test software against its specification but using some knowledge of its internal workings
- **GUI** – Graphic User Interface
- **Integration Testing** – Testing of combined parts of an application to determine if they function together correctly. Usually performed after unit and functional testing. This type of testing is especially relevant to client/server and distributed systems

- **ISO 9000** – A family of standards used for quality management systems. ISO stands for the International Organization for Standardization
- **Load Testing** – Testing that creates demand on a system to define its limits by measuring the system's response
- **Localization Testing** – Testing software specifically designed for a specific locality – this testing focuses on language
- **Metric** – A standard of measurement. Software metrics are the statistics describing the structure or content of a program. A metric should be a real objective measurement of something such as number of bugs per lines of code
- **Negative Testing** – Testing aimed at showing software does not work and checking for failure states
- **Performance Testing** – Testing conducted to evaluate the compliance of a system or component with specified performance requirements. Often this is performed using an automated test tool to simulate large number of users
- **Post Mortem** – An end-of-project wrap up meeting to review the project and set a path for improvement
- **Pre-Alpha** – A phase of testing during which builds that are only semi-functional. Testing focuses on specific features, not holistic testing
- **QA** – Quality Assurance

- **Quality Assurance** – All the planned or systematic actions necessary to provide adequate confidence that a product or service is of the type and quality needed and expected by the customer
- **Regression Testing** – Testing a previously tested program following modification to ensure that faults have not been introduced as a result of the changes made
- **Release Candidate** – A pre-release version, which contains the desired functionality of the final version, but which needs to be tested to verify it is ready for release
- **Sanity Testing** – Brief test of major functional elements of a piece of software to determine if it's basically operational (See also Smoke Testing)
- **SCM** – Software Configuration Management
- **SDLC** – Software Development Lifecycle or Software Design Lifecycle
- **Smoke Testing** – A quick-and-dirty test that the major functions of a piece of software work. The name of this test originated in the hardware testing practice of turning on a new piece of hardware for the first time and considering it a success if it does not catch on fire

- **Soak Testing** – Running a system at high load for a prolonged period of time. For example, running several times more transactions in an entire day (or night) than would be expected in a busy day, to identify and performance problems that appear after a large number of transactions have been executed
- **Software Development Lifecycle** – The processes and parameters that govern the creation of software in a defined project or projects
- **SQA** – Software Quality Assurance
- **Stress Testing** – Testing the limits of a system by altering the external variables
- **System Testing** – Testing that attempts to discover defects that are properties of the entire system rather than of its individual components
- **Test Bed** – An execution environment configured for testing. May consist of specific hardware, OS, network topology, configuration of the product under test, other application or system software, etc. The Test Plan for a project should enumerated the test beds(s) to be used
- **Test Case** – A commonly used term for a specific test

- **Test Environment** – The hardware and software environment in which tests will be run, and any other software with which the software under test interacts when under test including stubs and test drivers
- **Test Plan** – A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. Ref IEEE Std 829
- **Test Suite** – A collection of tests used to validate the behavior of a product
- **UI** – User Interface
- **Unit Testing** – Testing of individual software components. Often this testing is performed at the code level
- **Usability Testing** – Testing the ease with which users can learn and use a product
- **Use Case** – The specification of tests that are conducted from the end-user perspective. Use cases tend to focus on operating software as an end-user would conduct their day-to-day activities
- **Waterfall Testing** – A model that is sequential, flowing through several predefined phases
- **White Box Testing** – Testing based on an analysis of internal workings and structure of a piece of software

## Status Report Template...

**Tests Completed:**

**Priority Issues:**

**Current Projects:**

**Unscheduled Projects:**

**General Comments:**

~ 18 ~

**[www.Successful-Quality-Assurance.com](http://www.Successful-Quality-Assurance.com)**

## Bug Template...

---

**Brief Description:**

**Expanded Description:**

**Reproducibility:**

**Steps to Reproduce:**

- 1.
- 2.
- 3.
- 4.
- 5.

**Result:**

**Expected Result:**