# Acceptance Tests

**Test Name**:   
**Requirement(s) Tested**:   
**Outline**:   
**Pre-requisites**:   
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
|  |  |  |
|  |  |  |

**Test Name**: PD\_1  
**Requirement(s) Tested**: 1, 25  
**Outline**: Load a Protocol plugin  
**Pre-requisites**: Service is running, a protocol plugin is available for loading, CLI open  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter command to load protocol plugin | Service accepts command without issue |

**Test Name**: PD\_2  
**Requirement(s) Tested**: 2, 25  
**Outline**: Load a communication plugin  
**Pre-requisites**: Service is running, a communication plugin is available for loading, CLI open  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter command to load communication plugin | Service accepts command without issue |

**Test Name**: PD\_3  
**Requirement(s) Tested**: 3, 25, 29, 30, 31  
**Outline**: Program reads the maximum threads from the testcase file  
**Pre-requisites**: Service is running, a testcase is ready to read, CLI open  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter command to load a test | Service accepts command without issue |
| *2* | Read log | Log shows that maximum threads has been set |

**Test Name**: PD\_4  
**Requirement(s) Tested**: 3, 4, 25, 29, 30, 31  
**Outline**: Program reads the maximum threads from the testcase, applies hard limit when too high  
**Pre-requisites**: Service is running, a testcase is ready to read, CLI open  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter command to load a test | Service accepts command without issue |
| *2* | Read log | Log shows that maximum threads has been set (lower than asked for but *at* the limit) |

**Test Name**: PD\_5  
**Requirement(s) Tested**: 5, 6, 7, 8, 14, 15, 18, 19, 25, 27, 28  
**Outline**: While a test is running, measure performance changes when making other queries against the system  
**Pre-requisites**: Service is running, a testcase is running, CLI is open, log is being read  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter different commands to interact with the system | The system accepts the command |
| *2* | Do this several times in a row | The system happily performs each task |
| *3* | When the command completes, check logs | Logs should show no discernible difference in running time or efficiency |

**Test Name**: PD\_6  
**Requirement(s) Tested**: 9, 25  
**Outline**: When making changes regarding the threading facility, few files need changing, the ones that exist are relatively easy to edit  
**Pre-requisites**: Have source code, change needs to be made  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Look at area that needs changing | Only a single class needs editing |

**Test Name**: PD\_7  
**Requirement(s) Tested**: 10, 11, 25  
**Outline**: The service needs to be somewhat self-recoverable, and should be handled by sysvinit/system.  
**Pre-requisites**: Device off, software installed on device and setup as a service/daemon  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Turn system on |  |
| *2* | Check running services | ProtDev should be listed |

**Test Name**: PD\_8  
**Requirement(s) Tested**: 11, 20, 21, 22  
**Outline**: The service should be started exclusively as a service/daemon  
**Pre-requisites**: Service is not running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Attempt to run a ProtDev executable directly | Program should error out requiring it to be run as a service |

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Attempt to run ProtDev via the systems init infrastructure | The program should happily start running and the init systems logs should show a running service |

**Test Name**: PD\_9  
**Requirement(s) Tested**: 12, 25, 29, 32, 33, 34, 35, 36  
**Outline**: Initiate a test by passing in a testcase, it sends data to the target  
**Pre-requisites**: Service is running, protocol available, testcase written and correct, CLI open  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Send test case to program | Program accepts it without issue |
| *2* | Check logs | A log file for the test has been created, showing statistics and options provided along with any unexpected issues from the service |
| *3* | Check target | Target should show interaction with the service |

**Test Name**: PD\_10  
**Requirement(s) Tested**: 13, 14, 25  
**Outline**: Stop a test while it’s running  
**Pre-requisites**: Service is running, a test is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter a command to view running tests | A list is shown displaying tests IDs and their description |
| *2* | Enter a command referencing the test ID to stop it | Program accepts the command and begins to shut down the test cleanly |

**Test Name**: PD\_11  
**Requirement(s) Tested**: 15, 25  
**Outline**: As the program is running, query the program for its current statistics  
**Pre-requisites**:   
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter a command querying the current statistics for the programs running | The program outputs current stats, including any running threads and other such business |

**Test Name**: PD\_12  
**Requirement(s) Tested**: 16, 26, 45  
**Outline**: The API allows a UX designer access to the system for data or control purposes  
**Pre-requisites**: API is written and public  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Check documentation for API | API is present and made public for easy access |

**Test Name**: PD\_13  
**Requirement(s) Tested**: 17, 25  
**Outline**: Developer loads a plugin, program performs self-checks to verify the plugin is functioning correctly  
**Pre-requisites**: Service is running, plugin is written and ready to load  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter a command to solo load a plugin | Program accepts command without issue and loads the plugin |
| *2* | Wait | Program outputs that the plugin is working/valid |

**Test Name**: PD\_14  
**Requirement(s) Tested**: 23  
**Outline**: When the program is first launched performs a series of steps that validate its environment and rectify any issues that may be present  
**Pre-requisites**: Service hasn’t been started on this device before/clean installation  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Run either setup script or start the program | Program announces it’s the first time it’s being setup |
| *2* | Wait | Program will go through a checklist of items and setup its environment |
| *3* | Wait | Service will be running and will have set up its own directories with its required libs, logs and other such locations |

**Test Name**: PD\_15  
**Requirement(s) Tested**: 25, 55  
**Outline**: While the program is installed perform an uninstallation  
**Pre-requisites**: Service is installed and configured on the system  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Enter command to remove the service | The program begins exiting gracefully |
| *2* | Wait | The uninstallation process will verify what the user wishes to do with user edited directories (plugins, logs, configs) |
| *3* | Enter choices | The program will carry out the choices (remove or keep and tar) |
| *4* | Wait | The program will no longer have a significant presence on the device |

**Test Name**: PD\_16  
**Requirement(s) Tested**: 24, 25  
**Outline**: While the program is running/performing tests, perform a hard yet As-Graceful-As-Possible (AGAP) exit.  
**Pre-requisites**: Program is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Send the Programs kill command on the CLI | The program will begin attempting to kill all its child threads as quickly as possible while attempting to maintain some level of graceful exit |
| *2* | Verify resources returned and program killed | The program should no longer be present, and any resources returned to the system |

**Test Name**: PD\_17  
**Requirement(s) Tested**: 27  
**Outline**: Perform an API call to find out about running tests, no tests are running  
**Pre-requisites**: Program is running, no tests are running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Perform an API call | A list of running tests is displayed, the list is empty. |

**Test Name**: PD\_18  
**Requirement(s) Tested**: 27  
**Outline**: Perform an API call to find out about running tests. At least a single test is running  
**Pre-requisites**: Program is running, at least oen test is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Perform an API call | A list of running tests is displayed, the tests are displayed |

**Test Name**: PD\_19  
**Requirement(s) Tested**: 28  
**Outline**: Perform an API call to get data about the running software  
**Pre-requisites**: Software is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Perform an API call to get Software running information | A collection of data representing |

**Test Name**: PD\_20  
**Requirement(s) Tested**: 30  
**Outline:** The software tells the user where the syntax is incorrect  
**Pre-requisites**: Software is running, a syntactically incorrect test case is ready to be read in  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Send test case to software | Software acepts the testcase |
| *2* | Wait | Software will declare that the testcase is syntactically incorrect and print the line/location where it is wrong |

**Test Name**: PD\_21  
**Requirement(s) Tested**: 31  
**Outline**: Verify the correct number of threads have been created (the exact numebr wanted)  
**Pre-requisites**: Software is running and the software’s current number of threads is known and there’s enoguh space for more threads to be created.  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Check the number of running threads | There will be the exact number of extra threads requested in the test thread. |

**Test Name**: PD\_22  
**Requirement(s) Tested**: 31  
**Outline**: Verify the software limits the number of newly created threads appropriately  
**Pre-requisites**: Software is running and the software’s current number of threads is known and there’s too many threads to create the full roster required by the new test  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Check the number fo running threads | The number of threads will be limited to whatever the programs max is and logs its decision |

**Test Name**: PD\_23  
**Requirement(s) Tested**: 32, 58  
**Outline**: Verify the software correctly loads the protocol as defined within the testcase  
**Pre-requisites**: Software is running and there is a protocol ready for the software to read  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Wait | The software will continue running |

**Test Name**: PD\_24  
**Requirement(s) Tested**: 32  
**Outline**: Verify the software reports protocol loading failures appropriately  
**Pre-requisites**: Software is running, a testcase with an incorrect protocol is listed  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Wait | The software will state/write to the log that the desired protocol is missing/can’t be found/doesn’t exist |

**Test Name**: PD\_25  
**Requirement(s) Tested**: 33  
**Outline**: Verify the software roughly matches the desired traffic rate  
**Pre-requisites**: Software is running, a testcase is written  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Check the target for interaction | The target is logging transactions at roughly the desired rate |

**Test Name**: PD\_26  
**Requirement(s) Tested**: 34  
**Outline**: Verify that some amount of noticeable chaos is applied to the traffic  
**Pre-requisites**: Software is running, a testcase is written  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Check the target for interaction over an extended period | The target is logging some wild differences in received traffic |

**Test Name**: PD\_27  
**Requirement(s) Tested**: 34  
**Outline**: Verify that the chaos can indeed be 100% turned off  
**Pre-requisites**: Software is running, a testcase is written  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Start a test | The test starts |
| *2* | Check the target of interaction over an extended period | The target is logging no wildly mad differences in received traffic |

**Test Name**: PD\_28  
**Requirement(s) Tested**: 36  
**Outline**: While a test is running, verify that data is correctly being sent to the target  
**Pre-requisites**: Software is running, a test is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Check the target for test data/traffic | The target is observed to be receiving test data/traffic |

**Test Name**: PD\_29  
**Requirement(s) Tested**: 37, 44, 45, 46  
**Outline**: Record the interpreted result codes  
**Pre-requisites**: Software is running, test is running, database is up  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Check database for result codes against action taken | There are result codes listed with the taken action |
| *2* | Check the target for the traffic | The target’s transaction logs match with the database’s |

**Test Name**: PD\_30  
**Requirement(s) Tested**: 56, 57, 59  
**Outline**: While a test is running, verify the settings used  
**Pre-requisites**: Software is running, a test is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Request test running data from the software | The program will return all currently running tests and show that the |

**Test Name**: PD\_31  
**Requirement(s) Tested**: 38, 39, 40, 41, 42  
**Outline**: Logs are kept and recorded for all aspects of the system and these logs can be read by an operator  
**Pre-requisites**: System is running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Open the logs directory | It contains a collection of logs that are being added to |

**Test Name**: PD\_32  
**Requirement(s) Tested**: 43  
**Outline**: The log should be a very minor hinderance to the traffic generation, verify that it is not  
**Pre-requisites**: Software running  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Run a test with the logger disabled | The Software will perform as asked |
| *2* | When the last test is finished, tun another with the logger enabled | The software will perform more or less identically to the last run |

**Test Name**: PD\_33  
**Requirement(s) Tested**: 47  
**Outline**: Previous runs should be recorded for comparison  
**Pre-requisites**: System is running, multiple previous tests have been run  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Check database for previous runs | The database shows previous runs |

**Test Name:** PD\_34 **Requirement(s) Tested**: 50  
**Outline**: The database should be accessable, i.e. created  
**Pre-requisites**: Software has at least run before  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Try to access the database | The database ahs been creaetd and can therefore be accessed |

**Test Name**: PD\_35  
**Requirement(s) Tested**: 51, 52  
**Outline**: The software should have knowledge of many instances for more complex runs and handle test load balancing from any currntly used instance  
**Pre-requisites**: Software has seperate running instances that can communicate  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Check database for knowledge of other instances | The database shows that the instances are talking and have knowledge of eachother |
| *2* | Start a test configured for Wide-Area testing | The current isntance will contact other instances with the desired configuration for performing tests with |

**Test Name**: PD\_36  
**Requirement(s) Tested**: 53  
**Outline**: The software should be able to access results regardless of what node the user might currently be sat on  
**Pre-requisites**: Multiple, communicating, instances are currently running and all have databases  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Request data from oine instance regarding another instance | The requested instance’s data is displayed/returned |

**Test Name**: PD\_37  
**Requirement(s) Tested**: 54  
**Outline**: When using the software, the user should be largely agnostic about other running instances and the software itself should handle everything  
**Pre-requisites**: Multiple, communicating, instances are currently running and all have databases  
**Method**:

|  |  |  |
| --- | --- | --- |
| ***STEP*** | *Action* | *Expected Observation* |
| *1* | Perform any task that might incurr the use of a seperate instance | The software will gladly handle the request, and unless there is a major/fatal issue, the user will not be aware of what is going happening |