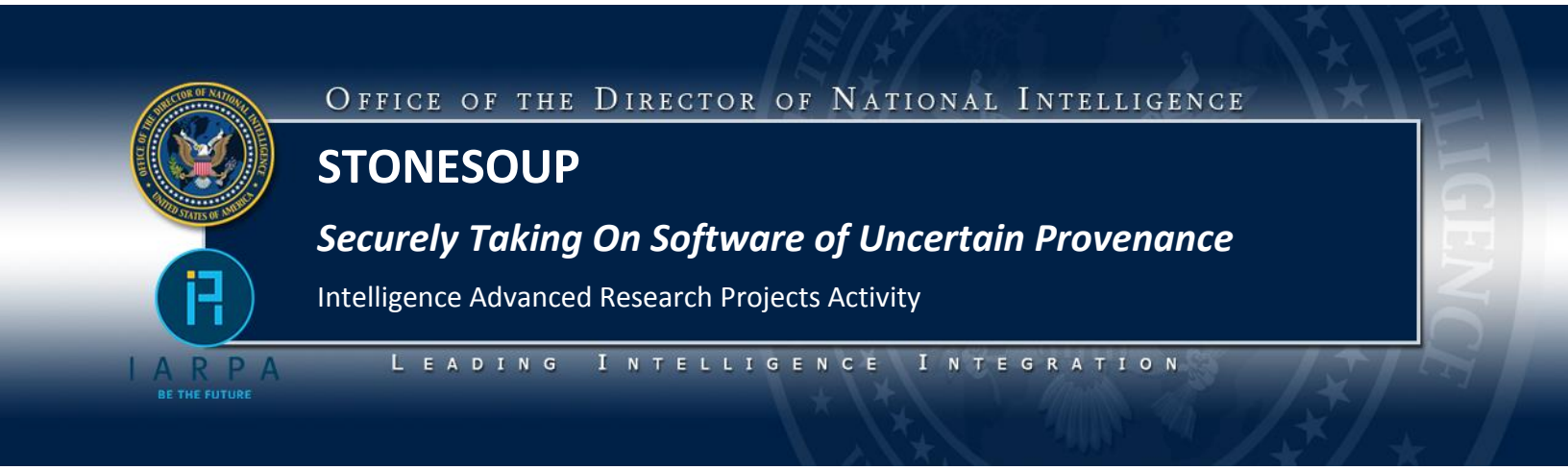

STONESOUP Phase 3
Test and Evaluation Execution and Analysis System (TEXAS)
Command Line Interface (CLI) User Guide
12 December 2014

This report was prepared by TASC, Inc., Ponte Technologies LLC, and i_SW LLC. Supported by the Intelligence Advanced Research Projects Activity (IARPA), Research Operational Support Environment (ROSE) contract number 2011-110902-00005-002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government.

Table of Contents

1 Overview

The Test & Evaluation eXecution and Analysis System (TEXAS) is designed and developed to test a Performer technology's ability to detect and mitigate software vulnerabilities and exploit through static analysis and run time countermeasures. To this end, the TEXAS system includes a Command Line Interface (CLI) to support the two major testing stages (i.e. Stage 1 and Stage 2) and the three major testing workflows (i.e. Analyze, Execute, and Score) within each stage.

The provided TEXAS CLI exists to allow a test engineer or performer technology developer to run the analyze or the execute commands, and to receive immediate feedback of the results and score, if performing an execution workflow. In addition to allowing execution of individual workflows, the TEXAS CLI also provides a validate command for convenience that performs both an analyze and execute in a single invocation.

The TEXAS CLI is deigned and develop to execute on a Linux operating system. While there are no specific Linux distribution requirements, all system testing has been performed against the following. Other distributions may also work, but are not specifically tested.

- Ubuntu 12.04 LTS (x86_64)
- CentOS 6.5 (x86_64)

Finally, TEXAS is implemented largely in Python, and requires a Python 2.7.5 execution environment. Due to the fact that not all Linux distributions provide a Python 2.7.5 package, Python 2.7.5 is built an installed from source as part of the dependency installation procedures.

2 Installation

2.1 Dependencies

The TEXAS system has dependencies on many 3rd party python modules. However, all of the additional module dependencies are codified into the installation process, and thus, automatically resolved and installed.

2.2 Installation Commands

Installation of the TEXAS CLI is performed via the defacto distribution process of python setuptools. There are two different mechanisms to execute the defined setup.py installation process, pip and setup.py. In both cases, TEXAS is installed to the site-packages folder of the current python environment. As part of installation, a default configuration file is written to ../../etc as well as the necessary scripts to ../../texas_scripts from the location of the Python executable. Successive runs of the installation procedure will update the configuration with new keys, values and sections but not overwrite existing values if they're non-default. It will also update the scripts except for any scripts starting with the 'performer_' prefix. It is important to note that TEXAS works well with python virtual environments (virtualenv). You can use the python virtualenv module to create a virtual environment, such that TEXAS does not interfere with your technology if you also rely on python scripts.

2.2.1 PIP

Pip is a tool for managing and installing Python packages. Installation and configuration details for Pip can be found at https://pypi.python.org/pypi/pip. Pip actually will download or unpack a given tarball and run the setup.py for the distribution package. Thus, to install with PIP, execute the following command.

```
pip install <texas_cli zip or tarball>
```

2.2.2 Setup.py

The TEXAS module includes a setup.py script which uses the Python setuptools module to actually do installation and configuration of the TEXAS CLI. To install via setup.py, extract the TEXAS release archive and execute the setup.py script via the following command.

```
python setup.py install
```

3 Configuration

The TEXAS CLI uses an INI style configuration file comprised of sections with key value pairs like those shown below. This style of configuration file is common to Linux programs and Python applications.

```
[section]

key=value
```

A TEXAS CLI has several sections, theses sections and their usage and purpose are explained in the following sections.

3.1 DEFAULT

The DEFAULT section of the TEXAS CLI configuration file provides critical information about different locations that are important to the TEXAS system, as well as information about what sections are database configuration sections. Each of these keys is required.

```
[DEFAULT]

workspace_dir=/opt/stonesoup/workspace  #base directory of the stone soup workspace

deps_dir=/opt/stonesoup/dependencies        # Path where all stone soup dependencies
will be stored

scripts_dir=/opt/stonesoup/texas_scripts   # Path where custom texas scripts are
stored

log_dir=logs    # Path where log files are stored defaults to ./logs

log_index_file=%(log_dir)s/index.pickle     # Path where reset scheme tracking is
stored

log_index_human_readable=%(log_index_file)s.txt  # Path to human redable log storage

databases=mysql,postgres      # Database types, that can be used for a given testcase

date_format=%%Y%%m%%d-%%H%%M%%S # Date format used, default is Y:M:D-H:M:S
```

3.2 commapi

The commapi section includes configuration options for the TEXAS Communications API Server including the IP address and port to listen on as well as a verbose flag for logging information about incoming and outgoing messages.

```
[commapi]

servip=127.0.0.1                # IP address that the Comms API Server Listens on

portnum=8886                    # Default port number the Comms API Server uses

verbose=False                   # Verbose Debugging of Comms API Server
```

3.3 analyze

The analyze section includes configuration options specific to the analysis step of a test case run.

```
[analyze]

command_file=build_commands                  # Name of the file inside the work space

logging_dir=%(log_dir)s/analyze              # Path to the Analyze logs, uses log_dir
as base

commapi_log=%(logging_dir)s/messages.xml  # Path to Comms API messages XML log

date_format=%%Y%%m%%d-%%H%%M%%S # Date format used, default is Y:M:D-H:M:S
```

3.4 execute

The execute section includes configuration options specific to the execute step of a test case run.

```
[execute]

command_file=run_commands                # Name of the file inside the work space

logging_dir=%(log_dir)s/execute          # Path to the execute log directory

commapi_log=%(logging_dir)s/messages.xml       # Path to the to Comms API messages XML
log

date_format=%%Y%%m%%d-%%H%%M%%S   # Date format used, default is Y:M:D-H:M:S

global_timeout=true                      # Respect the global time out value, Boolean

global_timeout_value=300                 # Time in seconds for global timeout of
execution

allow_reset_scheme=none,reboot,reprovision     # Allowed reset schemes, set to none
for no reboot

always_reboot=false                      # Disregard allow_reset_scheme and always
reboot
```

3.5 scoring

The execute section includes configuration options specific to the scoring step of a test case run.

```
[scoring]

logging_dir=%(log_dir)s/scoring                # Path to the scoring log files
```

3.6 performer

The performer section includes information about what type of performer is being run in a Stage 2 run of the TEXAS CLI.

```
[performer]

#performer tells us what type of analysis is being done

#and what type of binary distribution to use

#order comma delimited list (e.g. x86_64,x86)

#possible options are [ANY, x86, or x86_64]

performer=source,x86_64,x86,any
```

3.7 suffixes

The suffixes section includes information out the specific suffixes used in the naming of output archives from the analyze, execute and scoring steps of a test case.

```
[suffixes]

analyze=analyze            # analyze will be appended to all analyze log files

execute=result                    # result will be appended to all log files
```

3.8 environment_variables

The environment variables section provides the ability to pass arbitrary environment variables around to each of the processes executed by the TEXAS system. Environment variables included here will not override any of the specific environment variables used by the system.

```
[environment_variables]
<env_name>=<env_value>
```

3.9 Database Sections (mysql, postgres)

A database section is a section listed in the comma delimited list of database in the DEFAULT section. Each database section will have the same keys though their values may be different. TEXAS currently supports MySQL and Postgres for Database Processes. The first database of the appropriate type is what will be utilized when running a test case though it is allowable to have multiple.

```
[mysql]

abbr=mysql                  # Database name abbreviation name

dbtype=mysql                # Database type

host=ss-localhost           # Host-name or IP of database server

port=3306                   # Port of database

user=ss_db_user             # Database User Name

password=stonesoup2014      # Database User Password


[postgres]

abbr=pg                     # Database name abbreviation name

dbtype=postgresql           # Database type

host=ss-localhost           # Host-name or IP of database server

port=5432                   # Port of database

user=ss_db_user             # Database User Name

password=stonesoup2014      # Database User Password
```

3.10 Daemon Section

The Daemon Section is used to set configuration options for the automated test-case running of TEXAS test-cases. Jobs are pulled from a queue and actions are preformed based on their job type.

```
[daemon]

queue=ss_dev                    # Name of the queue to pull jobs from

job_types=analyze,execute       # Types of jobs machine will execute

outdir=/opt/stonesoup/daemon            # Directory to store workspace
```

3.11 LTTng Section

The (LTTNG) Linux Trace Toolkit – next generation. Section is used to configure the capture of performance metrics in side of an IO pair execution utilizing LTTNG and babeletrace (a LTTNG parser). It's important to note that babbletrace is not installed by TEXAS by default.

```
[lttng]

logging_dir=%(log_dir)s/execute/lttng          # Path where lttng log files will be
stored

babeltrace_path=%(deps_dir)s/bin/babeltrace   # Path to the babletrace executable
```

4 Usage

As with most Linux commands, the TEXAS CLI accepts a number of optional and require flags to provide configuration and data to the underlying workflow. The following sections provide additional descriptions of the command line sub-commands and flags. However, TEXAS will usage information if invoked with the "-h" or "—help" flags.

The TEXAS CLI is designed around the concept of sub-commands. Rather than provide multiple entry points to the TEXAS system, all of the available run options are provided as a sub-command to the TEXAS CLI. Many of the sub-commands share common run options, where possible, to streamline use.

4.1 General Overview

In an attempt to standardize usage of the TEXAS CLI, many run options are preserved across the available sub-commands. These include the following

Option	Usage	Description
-n, --dryrun	OPTIONAL	Run in do nothing mode
--release	OPTIONAL	Run in release mode
-l, --learning	OPTIONAL	Full path to initial learning archive"
-st, --state	OPTIONAL	Denote that Texas is loading from a saved state file. This is used for reset scheme
-c, --config	OPTIONAL	Instruct TEXAS to use the specified configuration file for this invocation. DEFAULT: ../../etc/texas/texas.conf (relative to the python executable)
-d, --debug	OPTIONAL	Instruct TEXAS to run in debug mode. This flag is passed down to the interfaces defined in the Communications API. It is mainly provided for performer use. DEFAULT: disabled
-o, --outdir	REQUIRED	Instruct TEXAS to write resulting output files to the specified directory.
-p, --performer	OPTIONAL	Instruct TEXAS to run with performer technology.
-s, --stop-on-error	OPTIONAL	Instruct TEXAS to stop on error, rather than continuing execution to the next test case (if more exist). DEFAULT: do NOT stop on error
-t, --testcases	REQUIRED	Provide a list of test cases to TEXAS for analysis, execution, or scoring. A minimum of one test case must be specified, but n are allowed.
-v, --verbose	OPTIONAL	Instruct TEXAS to run in verbose mode.

4.2 TEXAS Sub-Commands

The following TEXAS sub-commands are available to support analysis, execution, and scoring of target test cases.

4.2.1 Analyze

The TEXAS analyze sub-command runs the workflow to analyze a target test case in Stage 1 (i.e. without performer technology) or Stage 2 (i.e. with performer technology). The analyze command accepts a source or binary test case archive, and produces a new archive that is ready for execution on success. In the event of a failed build, an archive will be preserved for offline examination.

There are no special options or flags available or required during analysis that are not already defined in Section 4.1. An example invocation may take the form.

```
sudo texas_cli analyze –o <output_folder> -t <testcases archives>
```

4.2.2 Execute

The TEXAS execute sub-command runs the workflow to execute a previously analyzed target test case in Stage 1 (i.e. without performer technology) or Stage 2 (i.e. with performer technology). The execute command accepts an archive produced via the analysis command, and execute the target IO pairs. For each IO pair, a clean copy of the analysis archive is used to ensure that files do not cross-contaminate runs.

Once execute completes, the scoring checks will also be evaluated, and a final score produced. The entire test case workspace will be archived and preserved after each IO pair execution to support offline debugging.

In addition to the common options defined in Section 4.1, the following options are also supported for execute sub-command.

Option	Usage	Description
-io, --iopairs	OPTIONAL	Instruct TEXAS only run the specified IO pairs. The IO pairs are listed by name, as defined in the metadata. There are three special options to support running IO pairs of a particular category. These are: “all” – run all defined IO pairs “bad” – run only the bad IO pairs (if any) “good” – run only the good IO pairs (if any) DEFAULT: Run ALL defined in the metadata

Option	Usage	Description
-r, --result_file	REQUIRED	Instruct TEXAS to consolidate results in the target CSV file. The CSV file will contain a row for each output check, each formula, and the overall result for each IO pair. New results are always appended to the existing results form previous execution runs.

An example invocation may take the form.

```
sudo texas_cli execute –o <output_folder> -r <results_file> -t <testcases>
```

4.2.3 Validate

The TEXAS validate sub-command runs the both the analyze and execute workflows in sequence for a target test case in Stage 1 (i.e. without performer technology) or Stage 2 (i.e. with performer technology). This is a convenience sub-command to allow for the execution of the entire workflow in a single command. In the event that analyze was not successful, execute will be skipped. Both workflows are run as describe in the preceding sections.

In addition to the common options defined in Section 4.1, the following options are also supported for validate sub-command.

Option	Usage	Description
-io, --iopairs	OPTIONAL	Instruct TEXAS only run the specified IO pairs. The IO pairs are listed by name, as defined in the metadata. There are three special options to support running IO pairs of a particular category. These are: "all" – run all defined IO pairs "bad" – run only the bad IO pairs (if any) "good" – run only the good IO pairs (if any) DEFAULT: Run ALL defined in the metadata
-r, --result_file	REQUIRED	Instruct TEXAS to consolidate results in the target CSV file. The CSV file will contain a row for each output check, each formula, and the overall result for each IO pair. New results are always appended to the existing results form previous execution runs.

An example invocation may take the form.

```
sudo texas_cli validate –o <output_folder> -r <result_file> -t <testcases>
```

4.2.4 Daemon

The TEXAS daemon sub-command runs Texas in a daemon mode. This runs texas in an automated state where jobs are pulled down from a database and results are stored back on the database.

In addition to the common options defined in Section 4.1, the following options are also supported for daemon sub-command.

Option	Usage	Description
-q, --queue	OPTIONAL	Provide name of the Queue that Texas should pull jobs from

An example invocation may take the form.

```
sudo texas_cli perftest –o <output_folder> -r <result_file> -t <testcases>
```

4.2.5 Score

The TEXAS score sub-command runs the workflow to score a previously executed target test case in Stage 1 (i.e. without performer technology) or Stage 2 (i.e. with performer technology). The execute command accepts an archive produced via the execute command, and will re-evaluate the output checks and scoring formulas. This sub-command may be useful to manually debug perceived scoring issues or run new scoring checks if the metadata is updated.

In addition to the common options defined in Section 4.1, the following options are also supported for score sub-command.

Option	Usage	Description
-r, --result_file	REQUIRED	Instruct TEXAS to consolidate results in the target CSV file. The CSV file will contain a row for each output check, each formula, and the overall result for each IO pair. New results are always appended to the existing results from previous execution runs.An example invocation may take the form.

```
sudo texas_cli score –r <result_file> -t <testcases>
```

5 TEXAS Support Scripts

5.1 TEXAS Database Status Script

This status script will track analyze and execute process for a given queue when running the TEXAS daemon. This will provide you with percent complete as well as number of successful, failed, and skipped items.

Possible queue names are:

- **ss-ivv-kestrel**
- **ss-ivv-columbia**
- **ss-ivv-grammatech**

Option	Usage	Description
-i IP, --ipaddress IP	REQUIRED	IP address of database server
-p PORT, --port PORT	OPTIONAL	Port Number of database Server, Default 27017
-u Timer, --update TIMER	OPTIONAL	Timer in minutes to update the status view (Default 1min)
-q QUEUE_NAME, --queue QUEUE_NAME	REQUIRED	Name of the queue to build stats on

An example invocation may take the form.

```
sudo texasdb_status.py -i <ipaddress> -q <queue_name>Daemon Database Status Scripts
```

5.2 TEXAS Database Result Script

This status script will retrieve the scoring results from a given queue or all queues

Possible queue names are:

- **ss-ivv-kestrel**
- **ss-ivv-columbia**
- **ss-ivv-grammatech**
- **all**

Option	Usage	Description
-i IP, **--ipaddress IP**	REQUIRED	IP address of database server
-p PORT, **--port PORT**	OPTIONAL	Port Number of database Server, Default 27017
-v, **--verbose**	OPTIONAL	Run with output in verbose mode
-t, --type	OPTIONAL	Type of scoring to return, analyze, execute or both. Default is execute
-s, --status	OPTIONAL	Returns either completed, uncompleted or both, Default is completed
-q QUEUE_NAME, **--queue QUEUE_NAME**	OPTIONAL	Name of the queue to build stats on, Defaults to all if no queue name is provided

An example invocation may take the form.

```
sudo texasdb_result.py -i <ipaddress> -q <queue_name>
```

5.3 TEXAS Database Load Script

This database script will be used to load test cases into the daemon and activate a given queue.

Common Arguments

Option	Usage	Description
-c CONFIG, **--config CONFIG**	OPTIONAL	Load a non default configuration file
-v, **--verbose**	OPTIONAL	Run with output in verbose mode

Load Arguments

Option	Usage	Description
-t TESTCASE, **--testcase TESTCASE**	REQUIRED	Test-case Archive or Archives to process

An example invocation may take the form.

```
sudo texasdb_load load -v -t <TESTCASE NAME>
```

Activate Arguments

Option	Usage	Description
-i ID **--id ID**	OPTIONAL	ID of the queue to activate
-q QUEUE_NAME, **--queue QUEUE_NAME**	OPTIONAL	Name of the queue to build stats on, Defaults to all if no queue name is provided

An example invocation may take the form.

```
sudo texasdb_load activate -q <QUEUE_NAME>
```

5.4 TEXAS Database Hosts Script

Commands and scripts interact with machines running test-cases from TEXAS Daemon Database

Common Arguments

Option	Usage	Description
-i IP, **--ipaddress IP**	REQUIRED	IP address of database server
-p PORT, **--port PORT**	OPTIONAL	Port Number of database Server, Default 27017
-v, **--verbose**	OPTIONAL	Run with output in verbose mode

Activate Arguments

Option	Usage	Description
-m HOSTS, **--machines, HOSTS**	REQUIRED	Hostnames of the machines to activate on the database

An example invocation may take the form.

```
sudo texasdb_hosts.py activate –i <ipaddress> -m <host name>
```

Deactivate Arguments

Option	Usage	Description
-m HOSTS, --machines, HOSTS	REQUIRED	Hostnames of the machines to activate on the database

An example invocation may take the form.

```
sudo texasdb_hosts.py deactivate –i <ipaddress> -m <host name>
```

5.5 TEXAS Database Queue Script

A script for managing all TEXAS Daemon Queues

Common Arguments

Option	Usage	Description
-i IP, --ipaddress IP	REQUIRED	IP address of database server
-p PORT, --port PORT	OPTIONAL	Port Number of database Server, Default 27017
-v, --verbose	OPTIONAL	Run with output in verbose mode

Reset Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	REQUIRED	Queue name to reset all analyze and execute runs
-t TESTCASES, --testcases TESTCASES	OPTIONAL	Individual test-cases to reset on the database.(Default: All)"Only reset performer (stage 2) runs
-p, --performer	OPTIONAL	Only reset performer (stage 2) runs

An example invocation may take the form.

```
sudo texasdb_queue  -q <queue_name> -t <testcase>
```

Activate Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	REQUIRED	Queue name to reset all analyze and execute runs

An example invocation may take the form.

```
sudo texasdb_queue -q <queue_name>
```

Build Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	OPTIONAL	Queue name to reset all analyze and execute runs
-f, --force_insert	OPTIONAL	Force a test-case to be uploaded to a queue even if its already there.
-t TESTCASES, --testcases TESTCASES	OPTIONAL	Test-case or testcases to upload to the database
-io, IO_PAIRS, --io_paris IOPAIRS	OPTIONAL	Specific IOPairs for the queue
-p, --preformer	OPTIONAL	Run with Performer Technology
-d, --debug	OPTIONAL	Run in debug mode
-n, --dryrun	OPTIONAL	Run in do nothing mode
--release	OPTIONAL	Run in release mode
-s, --skip-if-exists	OPTIONAL	(NOT IMPLIMENTED) Skip testcases if they are already in the queue
--perf perf_runs	OPTIONAL	Number of runs to do for performance testing. Default: 1

An example invocation may take the form.

```
sudo texasdb_build -t <testcase> -q <queue_name> -d -n
```

Deactivate Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	REQUIRED	Queue name to reset all analyze and execute runs

An example invocation may take the form.

```
sudo texasdb_queue deactivate  -q <queue_name>
```

Delete Arguments

Option	Usage	Description
-i IP, **--ipaddress IP**	REQUIRED	IP address of database server
-p PORT, **--port PORT**	OPTIONAL	Port Number of database Server, Default 27017
-v, **--verbose**	OPTIONAL	Run with output in verbose mode

An example invocation may take the form.

```
sudo texasdb_queue delete –i <ipaddress> -q <queue_name>
```

Remove Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	REQUIRED	Queue to remove test-cases from
-t TESTCASES, **--testcases TESTCASES**	REQUIRED	Test-case or testcases to upload to the database

An example invocation may take the form.

```
sudo texasdb_queue remove  -q <queue_name> -t <testcase>
```

List Arguments

Option	Usage	Description
-q QUEUENAME, --queuename QUEUENAME	OPTIONAL	Queue to remove test-cases from
-o LOCATION, **--outdir LOCATION**	OPTIONAL	Location to store the file testcases.csv (Default is cwd)
-s STAGE, --stage STAGE	OPTIONAL	Stage wanted for each individual test-case. Choices (stage1, stage2, both)

An example invocation may take the form.

```
sudo texasdb_queue  list -q <queue_name> -o <location> -s <stage>
```

5.6 TEXAS Database Archive Script

A script for managing all TEXAS Daemon Queues

Common Arguments

Option	Usage	Description
-i IP, **--ipaddress IP**	REQUIRED	IP address of database server
-p PORT, **--port PORT**	OPTIONAL	Port Number of database Server, Default 27017

Testcase Arguments

Option	Usage	Description
-q QUEUENAME, **--queuename QUEUENAME**	REQUIRED	Queue name to pull the test-cases from
-t TESTCASES, **--testcases TESTCASES**	REQURED	Test-cases to pull form the database
-a, --analyze	OPTIONAL	Pull analyze runs from the test-cases
-e, --execute	OPTIONAL	Pull execute runs from the test-cases
--type TYPE	OPTIONAL	Type of test-cases to pull down. Choices (passed, failed, both)
-v, --verbose	OPTIONAL	Get verbose output
--orig ORGINAL, **--original ORGINAL**	OPTIONAL	Get the original test-cases
-o LOCATION, **--output-dir LOCATION**	OPTIONAL	Output directory to store the files at. Default is cwd
--uniq	OPTIONAL	Only get one instance of an iopair (Useful only with performer stage 2 runs)
-p, **--performer**	OPTIONAL	Pull down only performer (Stage 2) test-case runs.

An example invocation may take the form.

```
sudo texasdb_archive  -q <queue_name> -t <testcase> -p -v –type <passed, failed, both>
```