**CACIE Tool #NN** – ***Name***

**Version** **1.0**

**QA**: **TEST** or **NA** or **QA**

1. **Description and Purpose**

One or two paragraphs describing the tool’s function and purpose.

The Fingerprinter tool is a low-level utility that generates a 256-bit hash tag for an individual specified file or all files in a specified directory. The hash tag represents a unique value and hashes of two sets of data (i.e., two directories or two files) will match if and only if the corresponding data within the directories or files also match exactly. Small changes to the data within a file or directory will result in large, unpredictable changes in the hash.

1. **Functional Requirements**

The functional requirements of the tool will be documented in this section. Each requirement will have an ID, such as: FR-N, where N starts at 1 and increments for each Functional Requirement. Each of the Functional Requirement IDs will have a corresponding test ID listed in the RTM.

The following are the functional requirements of the Fingerprinter tool.

FR-1: Generate a hash tag for an individual file

FR-2: Generate a unique hash tag for a file directory and recursively generate a hash tag for each file contained within the file directory and its subdirectory(ies).

FR-3: Generate an output file that lists a datetime stamp and the path and filename and respective hash tag for a file or directory specified to be fingerprinted.

1. **Software Requirements Specifications**

The software requirements specification of the tool will be documented in this section.

Python 3.5

Python Standard Libraries:  
hashlib  
os  
argparse  
datetime

1. **Software Design Description**

The software design description of the tool will be documented in this section. The results of a Code Walkthrough with an independent third party will be summarized in this section.

The following is a brief description of the required arguments and the output generated by the Fingerprinter tool:

Positional Arguments:

* Target: file or directory path to be fingerprinted

Optional Arguments:

* -h, --help
* -o, --output OUTPUT [path and filename of output file; default is fingerprint.txt]
* Shell file configuration:

python [directory path]/pylib/fingerprint/fingerprint.py [optional arguments—see above] Name Arguments

* Output: The following information is logged to to a path and file (default, i.e. fingerprint.txt) or to a path and file, which is specified by user and is passed to the Tool Runner as an argument (-o/- -output OUTPUT)
* DateTime stamp
* Total number of files fingerprinted (if a directory is specified)
* Path and file name and corresponding hash for each file fingerprinted

1. **Requirements Traceability Matrix**

A requirements traceability matrix for the tool will be documented in this section. At a minimum, the matrix will include IDs of: Functional Requirements and the corresponding Acceptance Test, along with an indication of the test result (Pass/Fail).

The requirements traceability matrix for the Fingerprinter is presented in Table 1.

FR-1: Generate a hash tag for an individual file

FR-2: Generate a hash tag for a file directory and recursively generate a hash tag for each file contained in the subdirectory(ies) within the file directory.

FR-3: Generate an output file that lists the directories and files and their respective hash tags and a datetime stamp.

| **Table 1. Fingerprinter Tool Requirements Traceability Matrix** | | |
| --- | --- | --- |
| **Functional Requirement ID** | **Acceptance Test ID** | **Test Case** |
| QA Level | IT-1 | Installation Test |
| FR-1 FR-3 | ATC-1 | Fingerprint a file |
| FR-2 FR-3 | ATC-2 | Fingerprint a directory path |
| FR-1 FR-3 | ATC-3 | Fingerprint two files (identical except for filename) |
| FR-1 FR-3 | ATC-4 | Fingerprint two files (identical except for filename and date stamp) |
|  |  |  |

1. **Test Plan and Cases**

The test plan for the tool will be documented in this section. Each test will have a unique ID and criteria for determining if the test result is pass or fail. The TEST ID will be referenced in the RTM and ATR. An installation test, labeled **IT-1**, will be used by the Tool Runner to confirm the version of the tool being used is running correctly before launching it with the user’s parameters.

The Unit Testing done on the tool will be documented here, also.

| **Table 2. Fingerprint Test Plan** | | |
| --- | --- | --- |
| **TEST ID** | **Test Case** | **Test Result  (Pass/Fail)** |
| IT-1 | *Invoke Tool Runner and Fingerprint tool using runner\_ITC-1.sh* | |
| Verify Tool Runner is invoked and executes |  |
| Verify invoked tool executes |  |
| *Open git bash window in fingerprint\_test directory* | | |
| ATC-1 | *Enter the following command:*  *$ python fingerprint.py fingerprint.py -o fingerprint\_ATC-1.txt* | |
| Verify that the fingerprint\_ATC-1.txt documents datetime stamp of fingerprint, path (if applicable) and filename and hash tag |  |
| ATC-2 | *Enter the following command:*  *$ python fingerprint.py ./TestFolder\_1 -o fingerprint\_ATC-2.txt* | |
| Verify that the fingerprint\_ATC-2.txt documents datetime stamp of fingerprint, path (if applicable) and filenames and hash tag for each file in the directory and subdirectories |  |
| ATC-3 | *Copy and paste the file “fingerprint\_ATC-1.txt”*  *[directory now includes file “fingerprint\_ATC-1-copy.txt”]* | |
| *Enter the following command:*  *$ python fingerprint.py . fingerprint\_ATC-2.txt* | |
| Verify that the fingerprint\_ATC-2.txt documents identical hash tags for fingerprint\_ATC-1.txt and fingerprint\_ATC-1 - Copy.txt |  |
| ATC-4 | *Copy and paste “fingerprint\_ATC-1.txt”*  *[directory now includes file “fingerprint\_ATC-1-copy(2).txt”]* | |
| *Open fingerprint\_ATC-1-copy(2).txt in text editor—do not change text. Save file (i.e. change datestamp of file)* | |
| *Enter the following command:*  *$ python fingerprint.py . fingerprint\_ATC-4.txt* | |
| Verify that the fingerprint\_ATC-4.txt documents identical hash tags for fingerprint\_ATC-1.txt, fingerprint\_ATC-1 - Copy.txt, and *fingerprint\_ATC-1-copy(2).txt* |  |
| ATC-3 | *Navigate to the Test\_Repo\_Name repository and add a .txt file to directory* | |
| *Invoke Tool Runner and test tool using runner\_ATC-3.sh* | |
| Verify that runner\_ATC-3\_logfile.txt documents that the Tool Runner and test tool QA Status is TEST |  |
| *Delete addfile.txt file from directory* | |
| *Invoke Tool Runner and test tool using runner\_ATC-3.sh* | |
| Verify that Tool Runner QA Status is TEST  NOTE: Tool Runner is not on the approved tool list in the test repository—see \Test\_Repo\_Name\pylib\runner\config.json to verify |  |
| Verify that runner\_ATC-3\_logfile.txt documents that the invoked tool QA Status is QUALIFIED  NOTE: Invoked tool is on the approved tool list in the test repository—see \Test\_Repo\_Name\pylib\runner\config.json to verify | |
| *Open git bash window in Test\_Repo\_Name Repository* | |
| *Enter the following commands:*   * *git checkout development* | |
| *Invoke Tool Runner and test tool using runner\_ATC-3.sh* | |
| Verify that runner\_ATC-3\_logfile.txt documents that the Tool Runner and test tool QA Status is TEST |  |
| *Open git bash window in Test\_Repo\_Name Repository* | |
| *Enter the following commands:*   * *git checkout master* | |
| *ATC-4* | *Navigate to the Test\_Repo\_Name repository and add file addfile.txt to directory* | |
| *Open git bash window in Test\_Repo\_Name Repository* | |
| *Enter the following commands:*   * *git add addfile.txt* * *git commit ‘test commit’* | |
| *Invoke Tool Runner and test tool using runner\_ATC-4.sh* | |
| Verify that the Code Versions indicate that that local and remote repositories are not synced |  |
| Verify that runner\_ATC-4\_logfile.txt documents that the toolrunner and test tool QA Status is TEST |  |
| *Enter the following commands:*   * *git log -2* * *git reset –hard ‘first 6 characters of second commit SHA-1 hash’* | |
| *Invoke Tool Runner and test tool using runner\_ATC-4.sh* | |
| Verify that the local repository and remote repository versions are the same (no “not synced message”) |  |
| Verify that Tool Runner QA Status is TEST  NOTE: Tool Runner is not on the approved tool list in the test repository—see \Test\_Repo\_Name\pylib\runner\config.json to verify |  |
| Verify that runner\_ATC-1\_logfile.txt documents that the invoked tool QA Status is QUALIFIED  NOTE: Invoked tool is on the approved tool list in the test repository—see \Test\_Repo\_Name\pylib\runner\config.json to verify |  |

1. **Acceptance Test Report**

The test report will state whether the tool is qualified for use, summarize test case results, and report all resolved incidents and resolution of unresolved incidents.

1. **User Guide**

A guide for using the tool will be documented in this section.