**CACIE Tool #00** – ***Handprinter Tool (handprint.py)***

**Version** **1.0**

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

1. **Description and Purpose**

The Handprinter tool is a batch processor that processes a collection of files and folders by invoking the Fingerprinter tool on each one. An input file lists the collection of files and folders to fingerprint and optional filenames for the fingerprint. The Handprinter tool creates one fingerprint file for each item in the input file and stores the results in the target directory (default is current directory).

1. **Functional Requirements**

The following are the functional requirements of the Handprinter tool.

FR-1: Parse the files and/or folder listed in the input file.

FR-2: Invoke the Fingerprinter tool for each file and/or folder listed in the input file.

FR-3: Save fingerprint file to an optional filename as specified in the input file.

1. **Software Requirements Specifications**

The following documents the software requirements for the Handprinter tool.

Python 3.5

Python Standard Libraries:  
os  
sys  
argparse

Modules:  
.\fingerprint.py

1. **Software Design Description**

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

Positional Arguments:

* inputfile: Path to a file containing the items to process through the Fingerprinter tool. Each line should contain at least a path, and optionally a filename. Blank lines or those starting with a '#' are ignored. The path (separated by forward slashes) specifies the path to a file or folder to fingerprint. The optional filename specifies the name of the fingerprint file that is generated for the specified path.

Optional Arguments:

* -h, --help
* --outdir OUTDIR: Path to the folder where you want to store the Fingerprinter tool output. Defaults to the current directory.
* --sep, SEP: the type of character that separates columns in the input file. Defaults to a comma (,); specify a tab or a space delimiter as ‘s’

Shell file configuration:

python [directory path]/pylib/handprint/handprint.py [optional arguments—see above] inputfile

An independent code walkthrough was performed by Neil Powers on 11/22/2019. A summary and resolution of the comments is presented in Appendix A, table A-1.

1. **Requirements Traceability Matrix**

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

| **Table 1. Handprinter Tool Requirements Traceability Matrix** | | |
| --- | --- | --- |
| **Functional Requirement ID** | **Acceptance Test ID** | **Test Case** |
| QA Level | CACIE-handprint.py-IT -1 | Installation Test |
| FR-1 FR-2 | CACIE-handprint.py-TC-1 | Fingerprint files and/or directories listed in an input file |
| FR-3 | CACIE-handprint.py-TC-2 | Save fingerprint file(s) to fingerprint-optional\_filename.txt (NOTE: optional\_filename is specified in input file) |

1. **Test Plan and Cases**

The test plan for the Handprinter tool is presented in Table 2.

| **Table 2. Handprinter Tool Test Plan** | | | |
| --- | --- | --- | --- |
| **TEST ID** | **Test Case** | | **Test Result  (Pass/Fail)** |
| *Note [Testing\_Directory] in acceptance test report* | | | |
| *Navigate to [Testing Directory]\CA-CIE-Tools-Testing (code repository)* | | | |
| *Verify git branch and repository status by entering the following at the command line:*  *git branch (verify branch is “development”)*  *git pull*  *git status*  *git log -1*  *Note first 6 characters of git SHA-1 hash tag in acceptance test report* | | | |
| CACIE-handprint.py-IT-1 | *Linux platform:*  *In a Linux command window:*  *Navigate to [Testing\_Directory]\handprint\_test* | | |
| *Invoke Tool Runner and Handprint tool using handprint.py\_IT-1\_windows.sh by entering the following at the command line:./handprint\_ITC-1\_linux.sh* | | |
| Verify Tool Runner is invoked and executes |  | |
| Verify Handprinter tool executes |  | |
| *Windows platform:*  *In a Windows command window:*  *Navigate to [Testing\_Directory]\handprint\_test* | | |
| *Invoke Tool Runner and Handprint tool using handprint.py\_IT-1\_windows.sh by entering the following at the command line:./handprint\_ITC-1\_windows.sh* | | |
| Verify Tool Runner is invoked and executes | |  |
| Verify Handprinter tool executes | |  |
| CACIE-handprint.py-TC-1 | *Navigate to the [Testing\_Directory]\handprint\_test* | | |
| *Enter the following command:*  *./handprint.py-TC-1.sh* | | |
| Verify that the fingerprints outputted to *[Testing\_Directory]\ATC-1 correspond to the collection of files and directories listed in the input file ATC-1\_input.txt* | |  |
| CACIE-handprint.py-TC-2 | *Navigate to the [Testing\_Directory]\handprint\_test* | | |
| *Enter the following command:*  *./handprint.py-TC-2.sh* | | |
| Verify that the fingerprints outputted to [Testing\_Directory]\ATC-2 correspond to the collection of files and directories and the optional filenames listed in the input file ATC-2\_input.txt | |  |

1. **Acceptance Test Report**

Acceptance testing of the Handprinter tool was performed by Neira Mondragon and in accordance with the test plan documented in Section 6. The acceptance testing was performed in the following directory:

\\olive\backups\CAVE\sara-sandbox\ToolsTesting\handprint\_test

The Handprinter tool met the functional requirements documented in Section 2. Verification of the tool’s functionality is documented in Table A-2 and the test logs are included in Appendix A. There were no incidents requiring resolution and accordingly, there are no unresolved incidents.

1. **User Guide**

The Handprinter tool can be invoked from the command line using the arguments as specified in Section 4 (Software Design) and the arguments for the invoked tool. The Handprinter tool can also be invoked using the Tool Runner tool.

**Appendix A**

**Code Review Summary   
and   
Acceptance Testing Logs**

| **Table A-1. Handprinter Tool Code Review Summary** | | | | |
| --- | --- | --- | --- | --- |
| **Code Line** | **Comment** | **Function Impact** | **Suggested Change** | **Resolution** |
| 55,62 | Variable sep is treated like a global value for this sub function but as a passed as value to another subfunction | Readability issue | If treating as global value do it for both functions or pass it to both functions | Sep now treated as global in body of parse\_file() |
| 51 | Apply lower function to variable sep will eliminate issue with capitalized user inputs | Issue with capitalized user inputs | If lower(sep) = ‘s’: Or check user inputs at start of application. | Applied lower() |
| 57 | Consider adding a rstrip(separator) | If the file contains a separator at the end of a line you could wind up with an extra blank field | Consider adding a rstrip(separator) | Applied rstrip() |
| 100 | If a user uses the same output name, they will overwrite each other | As this is basically a batch process, I can see the user trying to have all of the fingerprints added to the same file. the way its currently written though each file in the batch would overwrite each other. | Add logic to prevent overwriting | Added logic in make\_handprint() to notify user when this happens |

**Figure A-1. Screenshots for ATC-1 and ATC-2 Confirmation**

**Test Log A-1. handprint.py\_IT-1\_linux Test Logs**

**Test Log A-2. handprint.py\_IT-1\_windows Test Logs**

**Test Log A-3. handprint.py\_TC-1 Test Logs**

**Test Log A-4. handprint.py\_TC-2 Test Logs**