**CACIE Tool #NN** – **Inventory Preprocessor Tool**

**Version** **1.0**

**QA**: **QA**

# Description and Purpose

The Inventory Preprocessor tool’s purpose is to create a comprehensive data set consisting of radionuclide and/or chemical aqueous volume releases as a function of time for Central Plateau sites. Solid waste releases (chemical and/or radionuclide) may optionally be included. The input file set consists of the following types:

1. **Site-Specific Inventory**: This dataset represents any site-specific information. Any number of site-specific sources may be included.
2. **Solid Waste Release**: The data set consists of modified (i.e., reduction of number of time steps via interpolation) output from solid waste release model(s).
3. **SIMv2 Release**: The data set consists of a single CSV file containing radionuclide and liquid inventory release estimates.
4. **Chemical Inventory Release**: The data set consists of a single CSV file containing chemical and liquid release estimates.
5. **SAC Liquid Release**: The data set consists of a text file (comma-delimited) containing estimates for liquid discharges. Only water releases are included from this source file.

The waste and liquid-only sites included in the comprehensive release data set are all part of the **VZEHSIT**, a compilation of waste sites and their vertices corresponding to their footprint boundary extents.

This tool’s specific task is to parse out the relevant information for the waste sites found in the **VZEHSIT** file to assemble a site list containing chemical/radionuclide and water releases over time.

# Functional Requirements

The following are the functional requirements (FR) of the Inventory Preprocessor Tool:

FR-1: Accept arguments at the command line. Arguments will include:

* Required: VZEHSIT file path (including file name)
* Optional: Chemical Inventory file path (including file name)
* Optional: SAC file path (including file name)
* Optional: Solid Waste Release directory path (including the directory name)
* Optional: Solid Waste Release file path (including file name)
* Optional: Site-Specific Inventory file path (including file name)
* Optional: SIMv2 file path (including file name)
* Optional: List of analytes to treat as chemicals: array of strings
* Optional: List of encodings to try when parsing input files: array of strings
* Optional: List of analytes to include in output file: array of strings
* Optional: When set to "True", SIMv2 entrained solids will be included in the output as liquid release(s): boolean
* Optional: Name for the output file: string
* Optional: When set to "True", output will be recorded in the "Legacy" output format. If set to "False", output will be in standard format: boolean
* Optional: The name of the log file: string
* Optional: Path of directory in which to save output files (including directory name)
* Optional: The number of significant digits to preserve in the output: integer
* Optional: List of possible names for site name columns: array of strings
* Optional: Adjusts the level of detail recorded in the log file: string
* Optional: List of possible names for water columns: array of strings
* Optional: List of possible names for year columns: array of strings

FR-2: Only sites whose site name is found in the **VZEHSIT** file will be included in the output

FR-3: Sites from the **SAC** are included if no other information is had from another input file. Sites with “241-“ in its site name are excluded. The exception to the “241-“ exclusion rule are sites with “241-C” in the site name (case-insensitive), which are included in the final output. Only water releases are considered from the **SAC**.

FR-4: The user may identify a list of analytes to process from the input files (including whether to process water or not).

FR-5: The user may identify a list of analytes to be treated as chemicals. The designation of whether an analyte is a chemical will determine whether the analyte(s) are parsed from the the **SIMv2** or **Chemical Inventory** files (no other input files are affected by the grouping). Designating analytes as chemicals also determines the formatting of the output headers for each analyte.

FR-6: If the “entrain solids” option is set to “true”, the tool will convert **SIMv2** records’ source type (e.g. “Solids” vs “Liquid”) to “Liquid” where the “Inventory Module” has the matching string “entrained” (case-insensitive).

FR-7: After parsing user-specified analytes from all input files provided (whose sites are found in the VZEHSIT list), the records are merged into a single file. The rules for merging the various input files into one output file are thus:

1. Records parsed from the **Site-Specific Inventory** file(s)
2. Records from **Solid Waste Release** file(s), excluding site(s) found in the **Site-Specific Inventory** file(s)
3. Records from **SIMv2**, excluding site(s) found in the **Site-Specific Inventory** file(s)
4. Records from **Chemical Inventory Release**, excluding site(s) found in the **Site-Specific Inventory** file(s)
5. Records from the **SAC** if site(s) not been listed in any other source

FR-8: When writing the output file, waste release information will be grouped on a site-by-site, analyte-by-analyte, and year-by-year basis

FR-9: The Source/Inventory Module column in the output will include which file(s) contributed to each site record for every year of waste release included, separated by an underscore character (e.g. “Chemical-Inventory\_SIMv2”).

FR-10: Supports a standard and a legacy mode. The two output modes are explicitly concerned with how to format the output file. If the “legacy\_mode” option is set to “false” then the output will reflect the “standard” formatting, and vice versa if the option is set to “true”.

FR-11: The user may specify the number of significant digits reported, the default is six significant digits.

FR-12: The user may modify the default file encoding to accommodate input files with special characters.

FR-13: The user may specify string patterns for input file columns corresponding to the site name, year, and water column. Multiple patterns may be supplied, allowing flexibility in the input files to use different naming conventions for their site name, year, and water columns.

FR-14: After compiling all of the information, the user-specified number of significant figures will be preserved, rounding to the final digit. The rounding method employed will always round up the final significant digit, always breaking ties in favor of the next-greater number. A tolerance of error of “one” is reserved for any given value at the final significant digit (e.g. 3.14159 +/-0.00001).

# Software Requirements Specifications

Version 3.6 of the Python programming language was used to develop this script. The libraries implemented by this tool consist of the following:

* argparse
* copy
* logging
* os
* pandas (version 0.24.2)
* re
* math
* pathlib

All but the “pandas” library are native to the Python v3.6 release. Additional software requirements are dependencies on upstream work products that are parsed by the Inventory Preprocessor tool. The work products of interest include the following files: **Site-Specific Inventory**, **Solid Waste Release**, **SIMv2 Release**, **Chemical Inventory Release**, and **SAC Liquid Release**. The dependencies in the case of these files refers to the formatting of each file which is described in section 4 under Input files.

# Software Design Description

Arguments:

The tool is executed from the command line in a terminal (Linux or Windows) in the following manner (positional argument numbers are explained below by the corresponding numbered list):

$ python inventory\_pp.py --VZEHSIT /path/to/VZEHSIT [other optional flags and input values]

The only required argument to run the script is the **VZEHSIT** file. However, providing this argument in isolation will not produce any output and will crash the script. The remaining arguments, though optional, may be combined to produce a single output with a selective set of inputs. Other flags will modify the output format or the parsing behavior. The optional arguments will be grouped into two categories: input files and auxiliary options.

The input file flags consist of the following (each flag is preceeded by two hyphen “-“ characters when entered at the command line:

| Table  Accepted User Arguments List | | | |
| --- | --- | --- | --- |
| **Flag** | **Group** | **Behavior** | **Usage Notes** |
| CHEMINV | Input File | **Optional: Chemical Inventory** file path (including the file name) |  |
| CLEANINV | Input File | **Optional: SAC** file path (including the file name) |  |
| RCASWR\_dir | Input File | **Optional: Solid Waste Release** directory path (including the directory name) |  |
| Site\_Specific | Input File | **Optional: Site-Specific Inventory** file path (including the name of the file) | May specify multiple files with this flag, e.g. --Site\_Specific ./file1.csv ./data/file2.csv |
| VZEHSIT | Input File | **Required: VZEHSIT** file path (including the name of the file) | **Requires** at least one of the other Optional input files to run correctly, as specified in the preceding section. |
| VZINV | Input File | **Optional: SIMv2** file path (including the name of the file) |  |
| chem\_copcs | Auxiliary | **Optional:** List of analytes to treat as chemicals | May specify multiple analytes with this flag, e.g. --chem\_copcs CN Cr U NO3  Default: CN, Cr, U, NO3 |
| codec\_list | Auxiliary | **Optional:** List of encodings to try when parsing input files | May specify multiple codecs with this flag, e.g. --codec\_list utf-8 iso-8859-1  Default Value: utf-8, iso-8859-1 |
| COPC | Auxiliary | **Optional:** List of analytes to include in output file | May specify multiple analytes with this flag, e.g. --COPC water h-3 sr-90 u cn  Default Value: WATER H-3 I-129 SR-90 TC-99 U CR NO3 CN |
| entrain\_sim\_solids | Auxiliary | **Optional:** When set to "True", **SIMv2** entrained solids will be included in the output as liquid release(s). | Default: True |
| ipp\_output | Auxiliary | **Optional:** Name for the output file | Default: preprocessed\_inventory.csv |
| legacy | Auxiliary | **Optional:** When set to "True", output will be recorded in the "Legacy" output format. If set to "False", output will be in standard format | Default: True |
| logger | Auxiliary | The name of the log file | Default: inventory\_pp.log |
| output | Auxiliary | **Optional:** Path of directory in which to save output files | Default: execution directory |
| sig\_figs | Auxiliary | **Optional:** The number of significant digits to preserve in the output | Default: 6 |
| site\_keys | Auxiliary | **Optional:** List of possible names for site name columns | May specify multiple site column names, e.g. --site\_keys site\_name "cie site name" "ca site name". If column names have spaces it is necessary to use double quotes around the column header name (single quotes will not work).  Default: SITE\_NAME “CIE SITE NAME”, “CA SITE NAME” |
| verbosity | Auxiliary | **Optional:** Adjusts the level of detail recorded in the log file, defaults to [ALL] | Default: ALL |
| water\_keys | Auxiliary | **Optional:** List of possible names for water columns | May specify multiple water column names, e.g. --water\_keys water volume liquid "volume mean [m3]"  Default: WATER, VOLUME, LIQUID, VOLUME MEAN [M3], VOLUME [M3] |
| year\_keys | Auxiliary | **Optional:** List of possible names for year columns | May specify multiple year column names, e.g. --year\_keys "discharge/decay-corrected year" YEAR  Default: DISCHARGE/DECAY-CORRECTED YEAR, YEAR |

Input Files:

All lines starting with a hashtag “#” will be considered comments and will not impact the parsing methods of the Inventory Preprocessor tool. All files are expected to be comma-delimited files.

The Inventory Preprocessor tool expects that all **VZEHSIT** file is a file with a single header line (skipped). Unique values (except for Null or empty strings) are taken from the first column of the file.

The Inventory Preprocessor tool expects that all **Site-Specific Inventory** file(s) have:

* A header line containing at least 3 columns (with corresponding rows of data in subsequent lines), the columns do not have to be in order (bracketed columns are descriptive of the type of column, not to be used verbatim, e.g. [Column])::
  + [Site name column]
  + [Year column]
  + [Analyte column] (may have multiple unique columns of analytes)

The Inventory Preprocessor tool expects that all **Solid Waste Release** file(s) have:

* A file name with a site name and analyte separated by an underscore (e.g. 200-E-30\_Sr-90.csv). File endings are irrelevant, but the file naming pattern must be [site name]\_[analyte].[file ending]
* Each file is expected to have 4 header lines (comments, with or without a hashtag) preceding the data header line (found on line 5).
* A header line (line 5) with 2 columns (exactly):
  + Reduced Year
  + Reduced Activity Release Rate (Ci/year)

The Inventory Preprocessor tool expects that all **SIMv2 Release** file(s) have:

* Requires three comment lines (not distinguished with hashtag characters), these lines are skipped by the tool
* The fourth line is the header line, expecting the following columns (bracketed columns are descriptive of the type of column, not to be used verbatim, e.g. [Column]):
  + Inventory Module
  + Source Type
  + [Site name column]
  + [Year column]
  + [Analyte Column] (one or more of these, including water)

The Inventory Preprocessor tool expects that all **Chemical Inventory Release** file(s) have:

* The fourth line is the header line, expecting the following columns (bracketed columns are descriptive of the type of column, not to be used verbatim, e.g. [Column]):
  + [Site name column]
  + [Year column]
  + [Analyte Column] (one or more of these, including water)

The Inventory Preprocessor tool expects that all **SAC Liquid Release** file is a comma separated file. The first line is a header line whose second column describes the number of waste sites in the file. Each line that has a waste site I the first column also has the number of condition changes in the second column. Each line after that has a year in the first column constitutes a “condition change”.

Output Files:

Two files are produced by this tool:

1. Preprocessed Inventory File
   1. If the legacy output option is set to “True”, the file will be a comma-delimited file with 11 header rows preceding the 12th row which contains the column names of the file. The 13th row contains the units for each corresponding column (if applicable). Fixed column names (regardless of the analytes selected) include: Inventory Module, SIMV2 site name, CA Site Name, Source Type
   2. If standard output is selected (legacy = “False”), then no leading header rows will be printed. Instead, the first row will contain the column names.
   3. The remainder of the file (in both legacy and standard output formats) is a combination of each input file and reflects the functional requirements described/tested in this document.
2. Log File
   1. Contains meta information printed/logged by the script processes. This information is not to be QA’d but is useful information for understanding the tool’s output.

Tool Runner:

The following is the shell script configuration that will be passed as an argument to the Tool Runner for qualified runs:

{directory path to repository}/\pylib\inventory\_pp\inventory\_pp.py “—VZEHSIT $VZEHSIT --\* $\*”

Each of these shell script variables (denoted by the “$”) will be set in the shell script with the corresponding variable input. The “--\*” and “$\*” symbols represent zero or more flags and corresponding shell script variables, depending on how the tool is executed

Code Review:

# Requirements Traceability Matrix

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

| Table  Requirements Traceability Matrix | | |
| --- | --- | --- |
| **Functional Requirement ID** | **Acceptance Test ID** | **Test Case** |
| QA Level | CACIE-cie-ipp.pl -IT-1 | Installation Test |
| FR-1 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will compare all site names of the output from cie-ipp.pl against **VZEHSIT**. If no differences are found (case differences are ignored), the checking script will print out “##QA-PASS (Waste Site Parse Check)”. |
| FR-2 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will compare all information from the **SIMv2 Release Inventory** file against the cie-ipp.pl output. If no differences are found, the checking script will print out “##QA-PASS (SIMV2 Check)”. |
| FR-3 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will compare all records from **Rerouted Inventory Releases** against the cie-ipp.pl output. If no differences are found, the checking script will print out “##QA-PASS (Rerouted Sites Check)”. |
| FR-4 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will compare all information from the **Chemical Inventory Release** files against the cie-ipp.pl output. If no differences are found, the checking script will print out “##QA-PASS (Chemical Inventory Check)”. |
| FR-5 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will verify that all records whose “Inventory Module” has the matching string “entrained” is listed as a “Liquid” waste type (relevant to the **SIMv2 Release Inventory**) in the cie-ipp.pl output. If no differences are found, the checking script will print out “##QA-PASS (SIMV2 Check)”. |
| FR-6 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will compare all records from the **SIMv2 Release Inventory** file (parsing only liquid inventory sources, including those records converted from solid to liquid in FR-5) against the cie-ipp.pl output. If no differences are found, the checking script will print out “##QA-PASS (SIMV2 Check)”. |
| FR-7 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will verify that **SAC Liquid Release** site water release information is included if no other sources provide waste release data for the site. The script will also verify that sites that have the character string “241-“ in the site name have been excluded with the exception of sites that have “241-C” in the name. If no deviations are found, the checking script will print out ‘##QA-PASS (SAC Check)”. |
| FR-8 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will verify that all waste release information has been grouped by site, copc, and year. If no variances are found, the checking script will print out ‘##QA-PASS (Comprehensive Check)”. |
| FR-9 | CACIE- cie-ipp.pl -TC-1 | Execute the checking script (common step for test case). The script will verify that all waste release information has been written accurately, rounding to the 6th significant digit. If no differences are found, the checking script will print out ‘##QA-PASS (Comprehensive Check in Reverse)”. |

# Installation Test Plan and Acceptance Test Plan Cases

The installation test plan for Inventory Preprocessor is presented in Table 2 and the acceptance test plan case for Inventory Preprocessor is presented in Table 3

| Table  **Inventory Preprocessor Installation Test Plan** | | | |
| --- | --- | --- | --- |
| **Inventory Preprocessor Installation Testing**  **CACIE-Inventory Preprocessor – IT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Tools Code Repository Directory: | | | |
| Navigate to the testing directory | | | |
| 1 | Invoke Tool runner and test the tool as follows:  *./runner\_run\_IT-1\_Inventory Preprocessor.sh* | | |
| 2 | Verify Tool Runner is invoked and executed. | Should see exact string: “QA Status: QUALIFIED : /opt/tools/pylib/runner/runner.py” |  |
| 3 | Error messages from the tool should also be recorded in the same log file. | Should see at least one line reading “Use of uninitialized value” |  |

| Table  **Inventory Preprocessor Acceptance Test Plan Case 1** | | | |
| --- | --- | --- | --- |
| **Inventory Preprocessor Acceptance Testing**  **CACIE-Inventory Preprocessor – AT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Inside of a Linux terminal, invoke the Tool Runner with the test input files as follows: *./run\_AT-1\_step-1.sh* | 5 files should be created:   1. ***runner\_run\_AT-1\_step-1.txt*** 2. ***step-1\_AT-1\_ipp.log*** 3. ***step-1\_AT-1\_ipp.csv*** |  |
| 2 | Open the ***runner\_run\_AT-1\_step-1.txt*** in a text editor. Look to verify that the runner tool is qualified and that the inventory preprocessor is “TEST” | The ***runner\_run\_AT-1\_step-1.txt*** should have 2 lines:   * QUALIFIED : /opt/tools/pylib/runner/runner.py * TEST: /home/[USER]/…/inventory\_pp.py   [USER] will be replaced by the user’s username and the ellipsis will be replaced by the full path to the file.  The key words are “QUALIFIED” and “TEST” as described. If present, this verifies that the tool and testing environment are valid. |  |
| 3 | Verify that the user arguments were accepted by comparing two files:   * ***run\_AT-1\_step-1.sh*** * ***step-1\_AT-1\_ipp.log*** | Open the files indicated in the test instruction column of this step and verify that each argument was accepted. A user verifies that each argument was accepted by ascertaining that what was specified in the shell script is printed within the first 20 lines of the log file.  Note: file paths specified in the shell script are relative paths or have the “~” shorthand. When accepted/written by the script to its log file the paths are written as absolute paths. |  |
| 4 | FR-2, FR-5, FR-6 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (SIMV2 Check)”.  If this exact string is present, then FR-2, FR-5, and FR-6 have been satisfied by the **Inventory Preprocessor** tool. |  |
| 5 | FR-3 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Rerouted Sites Check)”.  If this exact string is present, FR-3 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 6 | FR-4 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Chemical Inventory Check)”.  If this exact string is present, FR-4 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 7 | FR-7 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (SAC Check)”.  If this exact string is present, FR-7 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 8 | FR-8 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Comprehensive Check)”.  If this exact string is present, FR-8 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 9 | FR-9 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Comprehensive Check in Reverse)”.  If this exact string is present, FR-9 has been satisfied by the **Inventory Preprocessor** tool. |  |

# Acceptance Test Report

To complete the Acceptance Testing use Appendix A. This constitutes a single acceptance test that addresses all of the functional requirements listed in the traceability matrix.

Details of this test, when conducted, by whom, and if Passed or Failed are in Appendix A.

# User Guide

Refer to Section 4 of this software management plan for a full list of required inputs. It is recommended that a shell script be created to ease the burden of putting the command into a single command line argument. The recommended structure of this shell script is shown below:

TOOL=<path/to/cie-ipp.pl>

EHSIT=<path/to/waste/sites/file>

RADINV=<path/to/radionuclide/inventory/file>

CHEMINV=<path/to/chemical/inventory/file>

LIQINV=<path/to/liquid/inventory/file>

REDFIN=<path/to/file/containing/list/of/files/for/rerouting/information>

OUTPUT=preprocessed\_inventory

perl $TOOL $EHSIT $RADINV $CHEMINV $LIQINV $REDFIN $OUTPUT

# Tool Versions

This section details changes incorporated into each version of the **Inventory Preprocessor** tool.

* 1.0 – Tool was developed.

# References

IBM Knowledge Center, *Round-Half-to-Even Function*. Available at: <https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.xml.doc/doc/xqrfnrhe.html>.

# Appendix

**Completed Acceptance Test Cases**

**Testing Process Description**

The Inventory Preprocessor tool is checked using another script verifying only the first six functional requirements. The checking script is written in Python v3.6 and uses one library outside of the standard release called “Pandas”. The general process of the checking script is to first parse all of the input files into separate hashed dictionaries. The dictionaries are later combined into a final dictionary containing the expected result. Comparisons between the Inventory Preprocessor output (excluding the log and summary files) and the in-memory dictionary are made to verify compliance with the functional requirements. If the check passes, the script will print corresponding text to the log file for each functional requirement. In the event the checking script finds deviations from the functional requirements the output details which sites and waste streams were found to be out of compliance.

**Tool Runner Log**

| Table  **Inventory Preprocessor Acceptance Test Plan Case 1** | | | |
| --- | --- | --- | --- |
| **Inventory Preprocessor Acceptance Testing**  **CACIE-Inventory Preprocessor – AT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Inside of a Linux terminal, invoke the Tool Runner with the test input files as follows: *./runner\_run\_AT-1\_CIE-IPP.sh* | 5 files should be created:   1. ***runner\_run\_AT-1\_CIE-IPP\_log.txt*** 2. ***preprocessed\_inventory.csv*** 3. ***preprocessed\_inventory.log*** 4. ***preprocessed\_inventory-exclude.csv*** 5. ***preprocessed\_inventory-summary.csv*** |  |
| 2 | Inside of a Linux terminal, invoke the checking script: *./cie-ipp\_check.sh* | The script should produce a file called ***ipp\_check.log***. |  |
| 3 | FR-1 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Waste Site Parse Check): The cie-ipp.pl output only has sites listed in the VZEHSIT.”  If this exact string is present, FR-1 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 4 | FR-2, FR-5, FR-6 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (SIMV2 Check)”.  If this exact string is present, then FR-2, FR-5, and FR-6 have been satisfied by the **Inventory Preprocessor** tool. |  |
| 5 | FR-3 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Rerouted Sites Check)”.  If this exact string is present, FR-3 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 6 | FR-4 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Chemical Inventory Check)”.  If this exact string is present, FR-4 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 7 | FR-7 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (SAC Check)”.  If this exact string is present, FR-7 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 8 | FR-8 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Comprehensive Check)”.  If this exact string is present, FR-8 has been satisfied by the **Inventory Preprocessor** tool. |  |
| 9 | FR-9 Check (from ***ipp\_check.log***) | Open ***ipp\_check.log*** in a text editor and search for the string inside double quotes: “##QA-PASS (Comprehensive Check in Reverse)”.  If this exact string is present, FR-9 has been satisfied by the **Inventory Preprocessor** tool. |  |

# Appendix

**Completed Installation Test**

| Table B-1  **Inventory Preprocessor Installation Test Plan** | | | |
| --- | --- | --- | --- |
| **Inventory Preprocessor Installation Testing**  **CACIE-Inventory Preprocessor – IT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\inventory\_pp** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Tools Code Repository Directory: | | | |
| Navigate to the testing directory | | | |
| 1 | Invoke Tool runner and test the tool as follows:  *./runner\_run\_IT-1\_Inventory Preprocessor.sh* | | |
| 2 | Verify Tool Runner is invoked and executed. | Should see exact string: “QA Status: QUALIFIED : /opt/tools/pylib/runner/runner.py” |  |
| 3 | Error messages from the tool should also be recorded in the same log file. | Should see at least one line reading “Use of uninitialized value” |  |