**CACIE Tool #NN** – **CIE-IPP Tool**

**Version** **1.0**

**QA**: **QA**

# Description and Purpose

The CIE-IPP tool’s purpose is to create a comprehensive data set consisting of radionuclide and chemical aqueous volume releases as a function of time for Central Plateau sites. The data set consists of a modified compilation of a series of data sources, consisting of:

a. **SIMv2 Release**: The data set provides the radionuclide and liquid inventory release estimates from site inception to closure.

b. **Chemical Inventory Release**: The data set consists of a single CSV file containing waste releases for chemical waste releases, comprised of chromium, nitrate, uranium (total uranium), and cyanide.

c. **Rerouted Inventory Release:** The data set provides the rerouted chemical, radionuclide and liquid release originating from the U-10 and B-3 Pond sites.

d. **SAC Liquid Release**: The data set contain SAC liquid releases for non-contaminated effluent discharges.

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 VZEHSIT sites to assemble a site list containing chemical, radionuclide and water releases over time.

# Functional Requirements

The following are the functional requirements (FR) of the CIE-IPP Tool:

FR-1: Releases are included only if the site is part of the **VZEHSIT** data set

FR-2: Read in all **SIMv2 Release** records with temporal data

FR-3: Read in all **Rerouted Inventory Release** information, replacing or overriding other source(s) where applicable (on a site-by-site level)

FR-4: Read in all **Chemical Inventory Release** records with temporal data.

FR-5: Convert **SIMv2 Release** records’ source type (e.g. “Solids” vs “Liquid”) to “Liquid” where the “Inventory Module” has the matching string “entrained”

FR-6: Include only “liquid” type sources from **SIMv2** **Release** records (after satisfying FR-5)

FR-7: Read in the **SAC Liquid Release** only if the site record is not already included. For these sites, *only* the liquid release information is read in. This will also exclude sites that have the character string “241-“ in the site name. The exception to this exclusion rule are sites that have “241-C” in the name.

FR-8: All waste release information will be grouped on a site-by-site, copc-by-copc, and year-by-year basis

FR-9: After compiling all of the information, six significant figures will be preserved, rounding to the final digit. The rounding method follows the banker’s method, also known as the round-half-to-even method (IBM Knowledge Center), which will break ties between two values by favoring the value whose least significant digit is even. A tolerance of error of “one” is reserved for any given value at the 6th significant digit (e.g. 3.14159 +/-0.00001)

# Software Requirements Specifications

The Perl programming language was used to develop this script. No libraries outside of the standard distribution of the Perl v5.18.2 interpreter were used.

# Software Design Description

Arguments:

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

$ perl cie-ipp.pl [1] [2] [3] [4] [5] [6]

1. File path (including file name and extension) to the **VZEHSIT** data file
2. File path (including file name and extension) to the **SIMv2 Release** data file
3. File path (including file name and extension) to the **SAC Liquid Release** data file
4. File path (including file name and extension) to the **Chemical Inventory** **Release** file
5. File path (including file name and extension) to a file containing a newline-delimited list of file paths (each including the file name and extension) for the **Rerouted Inventory Release** information
6. Output filename and associated file path (not including the extension) for output file naming pattern and location

Input Files:

Each input file will be described in the same order as their corresponding positional argument for the script:

1. The waste site vertices file is a comma-separated file with a single header line (skipped). Unique values (except for Null or empty strings) are taken from the first column of the file.
2. The **SIMv2 Release** file is a comma-separated file with 4 header lines (2 are comment lines, the 3rd line in the 7th column is a unit descriptor). Column headings are taken from the 4th line in the file. The header row column rows are as follows:
   1. Inventory Module
   2. SIMV2 site name
   3. CA site name
   4. Source Type
   5. Volume [m3]
   6. Discharge/decay-corrected year
   7. C-14
   8. Cl-36
   9. H-3
   10. I-129
   11. Np-237
   12. Re-187
   13. Sr-90
   14. Tc-99
   15. U-232
   16. U-233
   17. U-234
   18. U-235
   19. U-236
   20. U-238
   21. Th-230 (decay only)
   22. Ra-226 (decay only)
3. The **Chemical Inventory Release** file is a comma-separated file with 1 header line. The column headings are as follows:
   1. Inventory Module
   2. SIMV2 Site Name
   3. CIE Site Name
   4. Source Type
   5. Year
   6. Volume Mean [m3]
   7. Cr [kg]
   8. NO3 [kg]
   9. U-Total [kg]
   10. CN [kg]
4. The **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 in 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”.
5. The newline-delimited list of rerouting information is a text file with Unix-style line endings (“\n”). Each line contains a single path to a file
6. This is not a file, but an argument indicating the base file name for the outputs.

Output Files:

Four files are produced by this tool:

1. <base file name>.csv
   1. This file is the final output. It is a comma-delimited file with 11 header rows containing meta information. The 12th row contains the column names of the file. The 13th row contains the units for each corresponding column (if applicable).
   2. The remainder of the file is a combination of each input file and reflects the functional requirements described/tested in this document.
2. <base file name>.log
   1. Contains more meta information printed/logged by the script processes. This information is not to be QA’d but is useful information for debugging the script.
3. <base file name>-exclude.csv
   1. This file is formatted the same as the final output, but it is a compilation of waste stream information that was excluded while parsing input files.
4. <base file name>-summary.csv
   1. Contains summary information of the total inventory by waste site and some breakdown of the integrated mass over specific time frames. This information, like the log file, is not to be QA’d but is useful information for evaluating the script and its 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}/\tools\cie-ipp \cie-ipp.pl “$EHSIT $RADINV $CHEMINV $LIQINV $REDFIN $OUTPUT”

Each of these shell script variables (denoted by the “$”) will be set in the shell script with the corresponding variable input.

Code Review:

Code walkthrough was performed by Jacob Fullerton on 09/17/2020. The code relies on a single bash/shell script file to execute the Perl script. No impacts to other repository tools or shared library dependencies were identified for the CIE-IPP tool.

# Requirements Traceability Matrix

The requirements traceability matrix for the CIE-IPP 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 CIE-IPP is presented in Table 2 and the acceptance test plan case for CIE-IPP is presented in Table 3

| Table  **CIE-IPP Installation Test Plan** | | | |
| --- | --- | --- | --- |
| **CIE-IPP Installation Testing**  **CACIE-CIE-IPP – IT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | | |
| **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\_CIE-IPP.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  **CIE-IPP Acceptance Test Plan Case 1** | | | |
| --- | --- | --- | --- |
| **CIE-IPP Acceptance Testing**  **CACIE-CIE-IPP – AT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | | |
| **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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 **CIE-IPP** 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 CIE-IPP 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 CIE-IPP 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  **CIE-IPP Acceptance Test Plan Case 1** | | | |
| --- | --- | --- | --- |
| **CIE-IPP Acceptance Testing**  **CACIE-CIE-IPP – AT-1** | | **Date: 9/18/2020** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | **Test Performed By: Dennis Fryar** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | | |
| **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*** | Pass |
| 2 | Inside of a Linux terminal, invoke the checking script: *./cie-ipp\_check.sh* | The script should produce a file called ***ipp\_check.log***. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |
| 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 **CIE-IPP** tool. | Pass |

# Appendix

**Completed Installation Test**

| Table B-1  **CIE-IPP Installation Test Plan** | | | |
| --- | --- | --- | --- |
| **CIE-IPP Installation Testing**  **CACIE-CIE-IPP – IT-1** | | **Date:** | |
| **Tool Runner Log File Location for this test:**  **\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | **Test Performed By:** | |
| **Testing Directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\cie-ipp** | | | |
| **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\_CIE-IPP.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” |  |

# Appendix C

**Completed Acceptance Test Cases**

|  |  |  |
| --- | --- | --- |
| **Table C-1. CIE-IPP**  **Code Review Summary** | | |
| **Code Line** | **Comment** | **Resolution** |
| ~100 | Although pared down to necessary functions for the CIE effort, cutting features from the base code (the CA IPP tool) means we have a new branch of code that is strictly for this application rather than an all-purpose tool. Because the tools’ functions/feature requirements are so similar, it is ill-advised to maintain two different tools that do nearly identical operations.  It is recommended that a future update merge the functional requirements of these two tools and produce outputs based on options or flags assigned by the user. | No modification required for tool release, to be addressed in long-term maintenance of product. |
| 101 | Hard-coding the accepted COPC list (similar comment for other aspects throughout the code) means that any change will need to be approved through a QA process. A better practice would be to make the code work with a list provided by the user, or use a default that can be overridden by the user. | No modification required for tool release, to be addressed in long-term maintenance of product. |
| General | The organization of the tool could take advantage of a more general approach. Right now each function duplicates code from other functions (particularly in parsing and merging records).  For efficiency and maintainability, it’d be better if there were a “parser” function that handled all of the files, then a “merger” function that could be used to merge records together. This would help streamline the code update process. | No modification required for tool release, to be addressed in long-term maintenance of product. |