Sandia National Laboratories
Carlsbad Programs Group

Waste Isolation Pilot Plant

Execution of Performance Assessment for the Compliance Recertification Application (CRA1)

Revision 0

ERMS# 530170

Author:	Jennifer Long (6821)	Jennifer Long	1/14/04
	Print	Signature	Date
Technical Review:	Brian Fox (6821)		1/14/04
	Print	Signature	Date
Management Review:	David Kessel (6821)	David Kessel	1/15/04
	Print	Signature	Date
QA Review:	Mario Chavez (6820)	Mario Chavez	1/15/04
	Print	Signature	Date

WIPP:1.4.1.3:PA:QA-L:530162

TABLE OF CONTENTS

LIST OF TABLES

1. Purpose

The purpose of this document is to describe the detailed process used for each of the major code specific calculations. A series of computer jobs, also known as runs, were run to support the first Compliance Recertification Application (CRA) of the Waste Isolation Pilot Plant (WIPP) during 2003, using WIPP Performance Assessment (PA) codes/models. All runs were made on the Sandia National Laboratories (SNL) HP Alpha Cluster. Heretofore, the Compliance Recertification Application will be referred to as CRA1.

Section 8 lists the steps that were followed to carry out the performance assessment. The analysis plan for these calculations is contained in SNL's Recertification Action Plan [1]. The information in this document provides a road map of the run control process and facilitates reproducibility/traceability for the CRA1 PA calculations. The recertification is required under the provisions of Title 40 CFR Part 191 Compliance Certification Application (CCA) for WIPP [2].

The first set of CRA calculations are referred to as the CRA1 calculations in the code management system (CMS) libraries (1 representing the first recertification effort). The calculations were run on a HP Alpha cluster running OpenVMS 7.3-1. The operating system and hardware of the WIPP PA Alpha Cluster have been upgraded since the CCA and PAVT calculations performed between 1995 and 1997 in order to stay abreast with the latest changes in software and hardware improvements. The upgrades were fully tested and found to have no impact on the PA codes.

2. Scope

This procedure applies to all official CRA1 jobs (code runs) executed on the HP Alpha Cluster. Each of the official jobs were executed utilizing scripts, with run-time input files and output files residing in an access controlled environment on the cluster. A script is a synonym for a DCL command procedure. All official jobs utilized the WIPP PA Software Configuration Management System (SCMS) to assure control of the various PA codes and their associated files.

The Run Control Coordinator coordinated and executed the code runs described in this document. The Coordinator's primary responsibilities were to:

- Complete all assigned code runs as described in this document.
- Provide scripted run procedures and the supporting environment that will enhance the reproducibility and traceability of each code run.

Code runs were excluded from this procedure if:

- They were not run by the Run-Control Coordinator, or
- They were not official code runs.

Note: The Run-Control Coordinator did not create or edit input code files used by the Code Sponsors/PA Analysts. Similarly, the Code Sponsors were not responsible for writing any portion of the Run-Control scripts and did not run any of the official calculations. All official CRA1 calculations were managed and executed solely by the Run-Control Coordinator.

The run control team consisted of:

Cliff Hansen, SNL, Team Lead for Total Systems Performance Assessment
Jennifer J. Long, SNL, Run Control Coordinator

The PA analysts included:

Joshua Stein, SNL, BRAGFLO, DBR and CUSP Code Sponsor
William P. Zelinski, RESPEC, BRAGFLO Supporting Analyst
Thomas Lowry, SNL, NUTS Code Sponsor
James W. Garner, PIRU, PANEL Code Sponsor
David Lord, SNL, DRSPALL Code Sponsor
Christi Leigh, SNL, EPAUNI, Code Sponsor
Joe Kanney, SNL, SECOTP, Code Sponsor
Bart Buell, GRAM, Database Control
Amy P. Gilkey, GRAM, CMS Code Build
Cliff Hansen, SNL, LHS, CCDFGF and CCDFSUM, Code Sponsor

3. Definitions

Access Control List (ACL)
Controls the type of access allowed (or denied) to a particular user or group of users. The user or group of users are represented with OpenVMS rights identifiers.

Alpha Cluster
A loosely coupled group of HP Alpha Systems, running the operating system OpenVMS 7.3-1, and configured as an OpenVMS cluster. This configuration allows for the sharing of resources (such as disk drives and batch queues) between the various nodes in the cluster.

CCA
Compliance Certification Application

CMS
Code Management System. A product from HP (formerly DEC) that was used to implement the Source Code Management System (SCMS) and is used for software configuration management and run control.

Code
A PA code package. May refer to the source code, or to the job that runs the code. Examples of codes include BRAGFLO, SECOTP2D, PANEL, etc.

Code Flow
The serial grouping of PA codes, in order of execution, to accomplish a given PA code calculation.

Code Run
See “Run”.

Code Sponsor
Person who oversees the QA process for a particular code.

Command Procedure
A program (script) written in Digital Command Language (DCL). The command procedure is used to control the environment, source of input files, code execution flow, and output file disposition for one or more codes.

CRA1
First Compliance Recertification Application

DCL
Digital Command Language. An interpretative programming language used to interact with the OpenVMS operating system. DCL commands can be embedded into command procedures, also called scripts.

EPA
Environmental Protection Agency

Job	See “Run”.
MMS	Module Management System by Compaq
PA	Performance Assessment.
PA Analyst	Person who interprets results for a particular code. The PA Analyst and the Code Sponsor can be the same person.
PAVT	Performance Assessment Verification Test
Reproducibility	The ability to re-run any individual calculation, as well as the overall group of calculations known as the CRA1 using the same inputs. The process needed to reproduce a calculation may be simple and straightforward, or very complex depending on numerous variables.
Run	An “instance” of a command procedure execution. That is, one execution of all series of codes required for a single calculation. For most codes an “instance” run represents a single vector.
Run Control Coordinator	A person whose role is to plan, script, run, audit, and document PA code execution.
SAN	Storage Area Network
Script	A synonym for a DCL command procedure.
SCMS	Source Code Management System. The SCMS is documented in the *SCMS Plan.*[3]
Traceability	The ability to demonstrate the steps executed during a code run, as well as, the pedigree of each component file used during each code run.

4. Preparation for the Calculation

The preparations for the CRA1 calculation fell into two categories. The first category included those tasks that closely duplicated tasks from the past performance assessments (CCA/PAVT/PAD/TBM); and include, identification of all input and output files, parameters/values, and output file disposition. The second category included creation/modification of run scripts, script input files, and creation of CRA1 CMS libraries/classes.

In terms of code flows, the CRA1 generally followed the same code flow as the CCA/PAVT assessments. All code flows, and the requirements for code input and output files, were decided by the PA Team Lead and provided to the Run Control Coordinator. Code input files were taken directly from the CCA/PAVT CMS libraries, except where parameter changes required the creation of a new version of an input file.

The EVAL scripts externalize as many run variables as possible. The variables included in the EVAL script input files include:

- Input file name and source.
- Output file name and disposition.
- Logical names. (i.e. working directory)
- Symbol names.
- Unique ID field.
- Analysis file copy indicator flag.
- Log file names and disposition.
- First code to execute.
- Last code to execute.

5. SNL WIPP HP Alpha System Architecture

The computer systems and applications architecture for the CRA1 calculations were required to be:

- Fully traceable and reproducible.
- Flexible enough to support tens of thousands of individual code runs.
- Extensible enough to handle modifications to code flows.
- Allow unofficial analysis runs.
- Manageable enough to allow hundreds of code runs, representing different parts of the overall calculation, to execute in the same time frame.
- Reliable enough to provide accurate schedule predictions to project management.

There are several key elements of the systems and application architecture that supports the CRA1 calculations. The most important elements to understand are:

- The HP Alpha cluster. This is a group of computers running version 7.3-1 of the OpenVMS operating system from HP Computer Corporation, now owned by Hewlett Packard Company. The WIPP computing cluster consists of two ES40's (4 processors each), two ES45's (4 processors each), an 8400 (12 processors), and a SAN that contains numerous disks that comprise over 2 terabytes of storage capability. Also included in the cluster is a single 2100 specifically set aside for the Environmental Protection Agency (EPA).
- CMS – A software product that has been used in past official PA's contributing to WIPP Certification. CMS is used for all aspects of source code management and run control. Objects controlled by CMS included Fortran sources, Fortran include files, MMS build scripts, production executables, code input files, EVAL script input files, EVAL scripts (RUN and RUN MASTER), and log files.
- EVAL run scripts. Each instance code run is controlled by an EVAL run script. The RUN script contains a set of commands for the correct code identification/imaging, file fetching, updating, creating, and inserting of files into the CMS. There may also be an EVAL RUN MASTER, which is used to control job submission/queue selection and allows a manageable approach to run control.
- EVAL input files. Each EVAL run script requires an EVAL input file that contains run specific information.
- Reliable controls on file access (security).

5.1. OpenVMS

Extensive documentation is available from HP Computer Corporation about OpenVMS. For the CRA1 calculations, an OpenVMS version 7.3-1 cluster consisting of 3 systems and 28 processors was utilized as described above. In a properly configured OpenVMS cluster, all disks are accessible from all cluster members. The cluster uses Storage Area Network (SAN) technology for efficient disk utilization and data storage/management. This allows for highly distributed processing, while providing for integrated data access.

5.2. CMS

A source control management system was implemented for the CCA and subsequent PAs using a product called CMS (Code Management System) from HP. The system proved to be extremely effective for addressing reproducibility and traceability issues, and was used for the CRA1. Basically, CMS is a utility that allows for the efficient storage and retrieval of multiple versions of files.

The CMS architecture used for past PAs will be used as the basis for the CMS architecture for the CRA1. New CMS libraries were created to store the calculation specific inputs and outputs for the calculation. Calculation classes, CRA1A and CRA1B, were created and all the relevant generations of all CRA1 files were included in the classes. Class CRA1A was used for all code runs up through CUTTINGS_S. All code runs after and including CUTTINGS_S used CRA1B. The reason for the CRA1B class was to separate the calculations that were already run without DRSPALL (class CRA1A) from the calculations that included the DRSPALL information (CRA1B).

A class is a set of specific element generations. It can be used to define a system version (such as a base level) consisting of different generations of several elements. An element generation can belong to zero, one, or several classes, but a class may contain no more than one generation of a given element. CMS libraries were created and DCL symbols were provided that followed the form LIBCRA1_XXX, where XXX was the code prefix (including secondary qualifiers) of interest. The term CRA1 is generic, and is intended to reflect that this was a calculation is the first recertification. It has no other meaning in this context.

5.3. Disk Drives

The HP Alpha cluster has a fixed number of disk drives which were used to store CMS libraries, user files, system files, executables, and other files needed by the operating system and the user community.

To provide for reproducibility and traceability, the Run Control Coordinator allocated specific disk drives (PAWORK and PACMS2) for exclusive use by the CRA1 PA calculations. Write level access to the CRA1 calculation and reference disks were restricted accordingly as determined by the ACL settings.

5.4. Batch Queues

The Run Control Coordinator created a set of batch queues for past PAs and were used again for CRA1. They support the distribution of jobs among the Alpha cluster. In this structure, a CCA batch queue was set up on each system, and one generic batch queue, CCA$BATCH, was set up to "feed" all the individual batch queues.

The node specific batch queues follow the naming convention: node_name$CCA, where node_name is the name of the system where the batch queue resides. So, CCR$CCA is the CCA calculation batch queue on node CCR. Only the Run Control Coordinator had write level access to these batch queues. The Run Control Coordinator had the ability to submit jobs directly to each node_name$CCA queue, or to the CCA$BATCH queue which would then forward the job to the first available execution queue.

5.5. EVAL Run Scripts

DCL command scripts, referred to here as EVAL run scripts, were used as the basis for all official code runs. The scripts are a series of embedded DCL commands that implement all code flows. As previously discussed, the run scripts from the PAVT were used as the basis for the CRA1 set of EVAL scripts. While most of the scripts were designed to be useable in both interactive and batch modes, official runs of instance scripts were done strictly in batch mode.

All EVAL scripts used in the CRA1 are stored in the CRA1_EVAL CMS library. The calculation scripts were created/maintained by the Run Control Coordinator. All CMS elements in the EVAL library conform to a naming convention imposed by the Run Control Coordinator. There are variations in the actual names due to file name length restrictions imposed by OpenVMS. The following naming conventions apply to scripts in the WP CMS library:

EVAL_CODE PREFIX_CRA1_RUN.COM	- instance script
EVAL_CODE PREFIX_CRA1_MASTER.COM.	- master script

Most of the code runs have at least two of the above script types.

The instance script performs one execution of a specific calculation or code flow; therefore all the individual codes that are part of the CRA1 calculation are executed in that single script. The instance scripts accept parameters that specify the EVAL script input file, EVAL script input file location, class, replicate, scenario, and vector values for the run.

When it exists, the master script is used to submit the runs of the instance script, typically by varying the vector number from 1 to 100. The master script prompts the user for replicate number, scenario number, and vector number ranges. Master scripts are essentially for the convenience of the Run Control Coordinator and do not directly impact any code runs.

When they exist, distribution scripts are run interactively. Distribution scripts prompt the user for ranges of values to be passed to the master script. These values are then used as parameters when the master script is submitted to a CCA batch queue for processing. Distribution scripts are essentially for the convenience of the coordinator and do not directly impact any code runs.

5.6. EVAL Run Script Input Files

All script variables have been removed from the run scripts and placed in the script input files. The ability of DCL to perform run-time symbolic substitutions allowed file names to be derived for each instance run dynamically. Each run script has at least one run script input file that specifies:

- Unique ID. This was CRA1 for all CRA1 runs. This field can be automatically inserted in files names allowing calculation specific file names to be generated with a minimal of effort.
- CMS class for all fetch activities.
- CMS class for all update activities.
- Default replicate, scenario, and vector numbers. During a calculation these variables are always replaced by command line values passed to the instance script.
- Log file name and post-run log file disposition.
- First code (in the predefined code sequence) to run. This allows code runs to be re-started at any step. The major use of this field in the CRA1 was to delineate "step1" runs, from "step2" runs. Step1 runs contain the codes that are run only once for some combination of replicate, scenario and vector. Step2 runs represent specific replicate, scenario, vector, and combinations.
- Last code (in the predefined code sequence) to run. The use of this field parallels of first code listed above, except is specifies the last code to execute in the sequence.
- Names and locations of all codes (executables) used during the run.
- Analysis directory and activity flag.
- Working directory.
- Code input files and their source
- Code output files and their post-run disposition.

- Any DCL symbols required by the run script. For example, the database view was always specified as a DCL symbol.
- Any DCL logical required by the run script. For example, the Analysis directory was always specified as a DCL logical.

Most of the variables in the script input file require run time substitution of other symbols from the same script input file. For example, the unique id, replicate, scenario, and vector values for the BRAGFLO output CDB file were defined at run time.

5.7. File Naming Conventions

The Run Control Coordinator attempted to embed as much meaningful information as possible in the names of files used during calculations, while still adhering to SCMS naming conventions. This was accomplished using a naming convention that provided the following information:

- The code associated with the file.
- The calculation type, i.e. BRAGFLO.
- An identifier indicating the file is part of the CRA1 calculations.
- The replicate number.
- The scenario number.
- The realization number or vector number, if applicable.
- The time intrusion value, if applicable.
- The cavity (upper, lower, or middle) used, if applicable.
- The mining type (full or partial) represented, if applicable.
- The file format or file type (input text, binary, .CDB, debug, etc.,)

Underscores (_) are normally used to separate the distinct elements of identification embedded in a file name. The first item in the file name was typically used to designate which code is reading or creating the file. The PA code prefix, defined in the SCMS Plan [3] is used as the designator. In some cases, the second item specifies the code prefix that a generic code is being run to support. The next item in the file name designates the calculation type or CCA code flow. Again, the prefix defined by the SCMS Plan is used for unique identification.

A file with the name GM_BF_CRA1.INP can be decoded as follows:

GM Stored in the GENMESH (GM) CMS library and related to the GM code.
BF This file relates to the code BRAGFLO (BF) run stream.
CRA1 This file relates to the name of the calculation.
INP This is a code input file.

Many files also include replicate, scenario, and vector references as follows:

R1 The replicate number associated with this calculation is “1”.
S3 The scenario number associated with this calculation is “3”.
V007 The vector number associated with this calculation is “007”.

6. Run Management and Verification

Management, tracking, and coordination of the CRA1 calculations was a time consuming operation, requiring 24-hour coverage, 7 days a week.

6.1. Queue Management

Batch jobs were controlled and distributed using a generic batch queue. Generic queues had the capability of moving jobs placed in the queue to compatible execution queues that had been previously defined. The list of batch queues available to the generic queue can be changed to include or exclude nodes on an as-needed basis. Submitting jobs to a generic batch queue provided the following benefits:

- One batch queue was used as the starting point for all runs,
- Stopping the generic queue effectively stops further submissions to CRA1 batch queues, which provided an easy mechanism for "pausing" job submissions,
- CRA1 batch queues pointed to by the generic queue could be stopped and started independently of the generic queue,
- Run management was simplified because job pending status was accomplished by examining one queue instead of multiple queues on multiple nodes.

6.2. Job Submission and Tracking

All CRA1 runs were submitted to execute on the HP Alpha cluster. There is no functional difference between runs on different nodes in the cluster. The Run Control Coordinator used a special OpenVMS account, CCA_MASTER, to submit all CRA1 runs to the Alpha cluster. This account is owned by the Run Control Coordinator and is only used for official runs and run validation. Access to the CCA_MASTER account is managed by the Run Control Coordinator.

Runs were submitted using the following two types of scripts:

- An instance script.
- A master script.

Where applicable, a master script was used to submit the instance script multiple times. The master script generally requires user-supplied parameters for replicate number, scenario number, vector number ranges, or any other parameters needed by the instance script. Master scripts eliminate the tedious and error-prone task of submitting hundreds and thousands of jobs manually.

The combination of master scripts and a generic batch queue provided a mechanism to easily submit multiple jobs to multiple queues. Jobs were monitored on the nodes where they were executing using OpenVMS utilities. Jobs not executing were held in a pending state on the generic batch queue. The pending jobs were also monitored with OpenVMS utilities.

6.3. Run Verification

The CRA1 runs were audited by the Run Control Coordinator to support the goals of reproducibility and traceability. The audit consisted of verifying that appropriate PA code output files were placed in CMS, verifying that all runs were completed, and verifying that log files were created for each run. The Run Control Coordinator did not generally have the requisite knowledge to validate the content of output files created by a PA code calculation. Therefore, no attempt was made by the Run Control Coordinator to characterize a run's success or failure based on output file contents.

6.4. Error Handling

The run scripts were designed and written to automatically notify the Run Control Coordinator when each run completed and to report final run status. OpenVMS mail messages were sent at run completion to the Run Control Coordinator. Each mail message provided the following information:

- Script parameters used as unique identification; this may have included replicate, scenario, vector, time intrusion, and/or cavity as applicable.
- Final disposition of the run; success or error.
- A list of PA codes run by the instance script with corresponding elapsed time.
- The log file name generated by the run.
- The disk location of the log file.

Errors generally fell into three categories:

- System resource conflicts.
- Script problems.
- Code problems.

When the error was due to a system resource conflict such as an inability to access a system authorization file, or a timing conflict between two concurrently executing jobs, the job was rerun. When the error was due to a problem in an EVAL run script, the Run

Control Coordinator corrected the script and the job was ran again. When the error was due to code problems the appropriate code sponsor corrected the problem.

6.5. Exception Handling

During the calculations issues periodically arose due to PA code runs where an error was reported by Run Control and verified by the respective PA analyst. Code Sponsors and PA Analysts subsequently determined a solution for each of these exceptions. Each solution was then codified via a modified run script input file (code configuration file). The PA code flow that generated an exception was then rerun using the appropriate special command procedure and the issue resolved.

Exception EVAL run script input files were derived from corresponding "regular" command procedure input files. Exception EVAL run script input files differed from regular EVAL run script input files in the following ways:

- PA code input files.
- PA code flow.

Where exception runs were required, a review was performed under the direction of the PA Team Lead. During the CRA1 two other types of changes were made to EVAL run script input files that addressed exception conditions. A modified script was prepared that addresses the different (modified) input file. In some instances, a special input file for a PA code was used. In other instances, PA code flow was changed; i.e. a PA code was eliminated from the exception run. Code Sponsors or PA Analysts determined which type of change was appropriate.

7. Code Versions

Table 7.1 lists the major codes and versions used in the CRA1 performance assessment.

Table 7-1 Code Versions Used in the CRA1 Calculation.

Code	Version	Build Date	Executable
ALGEBRACDB	2.35	31-JAN-1996	ALGEBRACDB_PA96.EXE
BRAGFLO	5.00	19-MAR-2003	BRAGFLO_QA0500.EXE
CCDFGF	5.00A	10-SEP-2003	CCDFGF_QA0500A.EXE
CCDFSUM	2.00	13-DEC-1996	CCDFSUM_PA96_2.EXE
CUTTINGS_S	5.10	24-OCT-2003	CUSP_QA0510.EXE
DRSPALL	1.00	06-OCT-2003	DRSPALL_QE0100.EXE
GENMESH	6.08	31-JAN-1996	GM_PA96.EXE
ICSET	2.22	1-FEB-1996	ICSET_PA96.EXE
LHS	2.41	6-MAR-1996	LHS_PA96.EXE
MATSET	9.10	29-NOV-2001	MATSET_QA0910.EXE
NUTS	2.05A	5-JAN-2001	NUTS_QA0205A.EXE
PANEL	4.02	20-MAR-2003	PANEL_QA0402.EXE
POSTBRAG	4.00	6-FEB-1996	POSTBRAG_PA96.EXE
POSTLHS	4.07	7-FEB-1996	POSTLHS_PA96.EXE
POSTSECOTP2D	1.04	5-JUN-1997	POSTSECOTP2D_QA0104.EXE
PREBRAG	7.00	19-MAR-2003	PREBRAG_QB0700.EXE
PRECCDFGF	1.00B	10-SEPT-2003	PRECCDFGF_QA0100B.EXE
PRELHS	2.30	27-NOV-2001	PRELHS_QA0230.EXE
PRESECOTP2D	1.22	12-JUN-1997	PRESECOTP2D_QA0122.EXE
RELATE	1.43	6-MAR-1996	RELATE_PA96.EXE
SECOTP2D	1.41A	09-JUL-2003	SECOTP2D_QA0141A.EXE
SUMMARIZE	2.20	11-JUL-1997	SUMMARIZE_QA0220.EXE

8. CRA1 Calculation Flow

This section documents the following information for each CRA1 calculation:

- calculation flow,
- database name where applicable,
- input/output files read/created,
- log files generated,
- command procedures used to run CRA1,
- command procedure input files,
- CMS libraries used for input files (shown in parentheses at right of file),
- CMS libraries used to store output files (shown in parentheses at right of file),
- and CMS libraries used to store log files.

NOTE: If files were intermediate and not kept in CMS, there will not be a library listed to the right of the file name.

In this section a short-hand notation is used to designate replicate numbers, scenario numbers and vector numbers. The notation follows:

Replicate number may be represented with R*x* where *x*= 1, 2, or 3
Replicate number may be represented with A*x* where *x*= 1, 2, or 3 (LHS only)
Scenario numbers are represented with S*y* where *y* = 1, 2, 3, 4, 5, or 6
Vector numbers are represented with V*nnn* where *nnn* can be = 001, 002, ... 100
Vector numbers are represented with *Rnnn* where *nnn* = 001, 002,...100 (LHS only)
Vector numbers are represented with V*v* where *v* = 1, 2, ... 100
Modified input files are represented with mo* where * = d, df, de, ds, dt
Cavity types are represented with *c* where *c*=U, M, L (U=Upper, M=Middle, L=Lower)
Mining types are represented with *M* where *M*= (FM) Full Mining and (PM) Partial Mining
Intrusion times are represented with int*i* where *i*=1, 3, 5, 7, 9
Intrusion time are represented with *ttttt* where *ttttt*=100, 350, 550, 750, 1000, 1200, 1400, 2000, 3000, 4000, 5000, 6000, 7000, 9000, 10000
Flow fields are represented with F*fff* where *fff*=001, 002, ... 100
Plot numbers are represented with pi_*nnn* where *nnn*=001, 002, 145

8.1. BRAGFLO

The BRAGFLO flow contains four distinct steps; step 1 includes GENMESH, MATSET, PRELHS, LHS, POSTLHS, ICSET, and ALGEBRACDB. Step 2 runs only PREBRAG, step 3 includes BRAGFLO and POSTBRAG, and step 4 ran only POSTALG. An exception step was added to handle any exception (modified vector inputs) and was run in a single step 3.

Step 1 is run once per replicate. Step 2 is run 100 times per replicate/scenario combination for a total of 1800 runs. Step 3 is run 100 times per replicate/scenario combination for a total of 1800 runs. Step 4 is run 100 times per replicate/scenario combination for a total of 1800 runs.

The BRAGFLO runs includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5, 6

Database Used:

Database: PARAMETER_PROD
The database is used by MATSET and PRELHS.

Log files:

BF_CRA1_R*x*_S*y*_STEP1.LOG
BF_CRA1_R*x*S*y*V*nnn*.LOG
BF_ALG2_CRA1_R*x*S*y*V*nnn*_STEP3.LOG (for both regular & exception runs).
BF_ALG2_CRA1_R*x*S*y*V*nnn*_STEP4.LOG

CMS Libraries for Log Files:

CRA1_BFR1S1 to CRA1_BFR1S6
CRA1_BFR2S1 to CRA1_BFR2S6
CRA1_BFR3S1 to CRA1_BFR3S6

Command Procedures:

EVAL_BF_CRA1_RUN.COM
EVAL_BF_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_BF_CRA1_STEP1.INP
EVAL_BF_CRA1_STEP2.INP
EVAL_BF_CRA1_STEP3.INP
EVAL_BF_CRA1_STEP3_GENERIC_MOD.INP
EVAL_BF_CRA1_STEP4.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.1.1. Step 1 Codes:

Step 1 codes are run using the EVAL_BF_CRA1_STEP1.INP command procedure input file.

GENMESH

Part of step 1 - run once per replicate.

Input files:

GM_BF_CRA1.INP	(CRA1_GM)

Output files:

GM_BF_CRA1.CDB	(CRA1_GM)
GM_BF_CRA1.DBG	

MATSET

Part of step 1 - run once per replicate.

Input files:

MS_BF_CRA1.INP	(CRA1_MS)

Output files:

MS_BF_CRA1.CDB	(CRA1_MS)
MS_BF_CRA1.DBG	

PRELHS

Part of step 1- run once per replicate. Sets a new random seed per replicate.

Input files:

LHS1_CRA1_A*x*.INP	(CRA1_LHS)

Output files:

LHS1_CRA1_TRN_A*x*.OUT	(CRA1_LHS)
LHS1_CRA1_A*x*.OUT	(CRA1_LHS)

LHS

Part of step 1- run once per replicate.

Input files:

LHS1_CRA1_TRN_A*x*.OUT	(CRA1_LHS)

Output files:

LHS2_CRA1_TRN_A*x*.OUT	(CRA1_LHS)
LHS2_CRA1_DBG_A*x*.OUT	(CRA1_LHS)

POSTLHS

Part of step 1- run once per replicate.

Input files:

LHS3_CRA1.INP	(CRA1_LHS)
LHS2_CRA1_TRN_A*x*.OUT	(CRA1_LHS)
MS_BF_CRA1.CDB	(CRA1_MS)

Output files:

LHS3_CRA1_A*x*_R*nnn*.CDB	(CRA1_LHS)
LHS3_CRA1.DBG	
LHS3_CRA11.SCR	
LHS3_CRA12.SCR	

ICSET

Part of step 1- run 100 times (vectors 1-100) per replicate.

Input files:

IC_BF_CRA1.INP (CRA1_IC)
LHS3_CRA1_A*x*_R*nnn*.CDB

Output files:

IC_BF_CRA1_R*x*_V*nnn*.CDB
IC_BF_CRA1_R*x*_V*nnn*.DBG

ALGEBRACDB

Part of step 1- run 100 times (vectors 1-100) per replicate.

Input files:

ALG1_BF_CRA1.INP (CRA1_ALG)
IC_BF_CRA1_R*x*_V*nnn*.CDB

Output files:

ALG1_BF_CRA1_R*x*_V*nnn*.CDB (CRA1_ALG)
ALG1_BF_CRA1_R*x*_V*nnn*.DBG

8.1.2. Step 2 Codes:

Step 2 codes are run using the EVAL_BF_CRA1_STEP2.INP command procedure input file.

PREBRAGFLO

Part of step 2 - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF1_CRA1_S*y*.INP (CRA1_BF)
ALG1_BF_CRA1_R*x*_V*nnn*.CDB (CRA1_ALG)

Output files:

BF2_CRA1_R*x*_S*y*_V*nnn*.INP (CRA1_BFR*x*S*y*)
BF1_CRA1_R*x*_S*y*_V*nnn*.DBG

8.1.3. Step 3 Codes:

Step 3 codes are run using the EVAL_BF_CRA1_STEP3.INP command procedure input file.

BRAGFLO

Part of step 2 - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF2_CRA1_R*x*_S*y*_V*nnn*.INP
BF2_CRA1_CLOSURE.DAT (CRA1_BF)

Output files:

BF2_CRA1_R*x*_S*y*_V*nnn*.OUT
BF2_CRA1_R*x*_S*y*_V*nnn*.SUM
BF2_CRA1_R*x*_S*y*_V*nnn*.BIN
BF2_CRA1_R*x*_S*y*_V*nnn*.ROT
BF2_CRA1_R*x*_S*y*_V*nnn*.RIN

POSTBRAGFLO

Part of step 2 - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF2_CRA1_R*x*_S*y*_V*nnn*.BIN
ALG1_BF_CRA1_R*x*_V*nnn*.CDB (CRA1_ALG)

Output files:

BF3_CRA1_R*x*_S*y*_V*nnn*.CDB (CRA1_BFR*x*S*y*)
BF3_CRA1_R*x*_S*y*_V*nnn*.DBG

8.1.4. Step 4 Code:

Step 4 codes are run using the EVAL_BF_CRA1_STEP4.INP command procedure input file.

ALGEBRACDB_2 (BRAGFLO POSTALG)

Step 4 - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
ALG2_BF_CRA1.INP	(CRA1_ALG)

Output files:

ALG2_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
ALG2_BF_CRA1_R*x*S*y*V*nnn*.DBG	

8.1.5. Step 3_4 Exception Runs:

BRAGFLO

Part of exception cases - run once per selected vector/ replicate/scenario combination using the run_master to select replicate/scenario/vector individually.

Input files:

BF2_CRA1_R*x*_S*y*_V*nnn*_MOD.INP	(CRA1_BFR*x*S*y*)
BF2_CRA1_CLOSURE.DAT	(CRA1_BF)

Output files:

BF2_CRA1_R*x*_S*y*_V*nnn*.OUT
BF2_CRA1_R*x*_S*y*_V*nnn*.SUM
BF2_CRA1_R*x*_S*y*_V*nnn*.BIN
BF2_CRA1_R*x*_S*y*_V*nnn*.ROT
BF2_CRA1_R*x*_S*y*_V*nnn*.RIN

POSTBRAGFLO

Part of step 3 (exception) - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF2_CRA1_R*x*_S*y*_V*nnn*.BIN	
ALG1_BF_CRA1_R*x*_V*nnn*.CDB	(CRA1_ALG)

Output files:

BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
BF3_CRA1_R*x*_S*y*_V*nnn*.DBG	

ALGEBRACDB (BRAGFLO POSTALG)

Part of step 3(exception) - run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
ALG2_BF_CRA1.INP	(CRA1_ALG)

Output files:

ALG2_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
ALG2_BF_CRA1_R*x*S*y*V*nnn*.DBG	

The EXCEPTION runs result in a log BF_ALG2_CRA1_R*x*S*y*V*nnn*_STEP3.LOG as the codes are run from BRAGFLO to POSTALG. Nothing except the input file to BRAGFLO distinguishes the exception run from any other BRAGFLO run. Table 8-1 lists the vectors that required exceptions runs by replicate/scenario.

Table 8-1 Exception runs by Replicate/Scenario

		Scenario					
Replicate		1	2	3	4	5	6
1	Vectors	2,90,98	29,79,98	2,11,98	79,98	2,18,29,98	2,29,98
2	Vectors	13,20,56,70, 81	56,72	13,56,70,81,99	56,71	13,56,70, 81	56,81,99
3	Vectors	3,20,27,28, 39,74,87,94	3,87	3,9,87	3,27,87	3,87	3,27,28,87

8.2. NUTS

8.2.1. NUTS (Screening)

NUT screening is run 100 times per replicate/scenario combination for a total of 1500 runs. The following replicates and scenarios are included:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Log files:

NUT_CRA1_SCN_R*x*S*y*V*nnn*_STEP1.LOG

CMS Libraries for Log Files:

CRA1_NUTR1S1 to CRA1_NUTR1S5
CRA1_NUTR2S1 to CRA1_NUTR2S5
CRA1_NUTR3S1 to CRA1_NUTR3S5

Command Procedures:

EVAL_NUT_CRA1_SCN_RUN.COM
EVAL_NUT_CRA1_SCN_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_NUT_CRA1_SCN_STEP1.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.2.1.1.Step 1 Codes:

The following code is run using the EVAL_NUT_CRA1_SCN_STEP1.INP command procedure input file.

NUT

Run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

NUT_CRA1_SCN_R*x*_S*y*.INP	(CRA1_NUT)
BF2_CRA1_R*x*_S*y*_V*nnn*.INP	(CRA1_BFR*x*S*y*)
BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)

Output files:

NUT_CRA1_SCN_R*x*_S*y*_V*nnn*.CDB	(CRA1_NUTR*x*S*y*)
NUT_CRA1_SCN_R*x*_S*y*_V*nnn*.OUT	

ALGEBRACDB

Run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

NUT_CRA1_SCN_R*x*_S*y*_V*nnn*.CDB	(CRA1_NUTR*x*S*y*)
ALG_NUT_CRA1_SCN_R*x*_S*y*.INP	(CRA1_NUT)

Output files:

ALG_NUT_CRA1_SCN_R*x*_S*y*_V*nnn*.CDB	(CRA1_NUTR*x*S*y*)
ALG_NUT_CRA1_SCN_R*x*_S*y*_V*nnn*.DBG	

8.2.2. NUTS (Screening Analysis)

NUT screening analysis is run 100 times per replicate/scenario combination for a total of 1500 runs. The PA Analyst will produce a list of screened-in vectors after the runs. The following replicates and scenarios are included:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Replicates 1-3, scenarios 1-5 will be run using SUMMARIZE_QA0220.EXE.

Log files:

NUT_CRA1_SCN_R*x*S*y*_STEP2.LOG

CMS Libraries for Log Files:

CRA1_NUTR1S1 to CRA1_NUTR1S5
CRA1_NUTR2S1 to CRA1_NUTR2S5
CRA1_NUTR3S1 to CRA1_NUTR3S5

Command Procedures:

EVAL_NUT_CRA1_SCN_RUN.COM
EVAL_NUT_CRA1_SCN_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_NUT_CRA1_SCN_STEP2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.2.2.1.Step 2 Code:

The following codes are run using the EVAL_NUT_CRA1_SCN_STEP2.INP command procedure input files.

SUMMARIZE

Run once per replicate/scenario combination.

Input files:

SUM_NUT_CRA1_SCN_R*x*_S*y*.INP (CRA1_SUM)
ALG_NUT_CRA1_SCN_R*x*_S*y*_V001.CDB*
through
ALG_NUT_CRA1_SCN_R*x*_S*y*_V100.CDB*

*Referenced in SUM_NUT_CRA1_SCN_R*x*_S*y*.INP files which contain names of input data files used by SUMMARIZE.

Output files:

SUM_NUT_CRA1_SCN_R*x*_S*y*.DAT (CRA1_SUM)
SUM_NUT_CRA1_SCN_R*x*_S*y*.LOG
SUM_NUT_CRA1_SCN_R*x*_S*y*_ERROR.LOG

8.2.3. NUTS (Non-Screening (ISO))

The NUTS non-screening step is run once for each replicate/scenario/vector combination listed below. The total number of vectors by scenario is not known until the screening runs (NUTS SCN Step 1 and Step 2) are made which results in the listing below which is provided by the NUTS analyst. The vectors in S1 with an asterisk (*) create "initialization" files for other non-screening runs (S2 to S5). Table 8-2 lists the screened vectors as shown by replicate/scenarios.

Table 8-2 Screened Vectors by Replicate/Scenario

Replicate	Scenario	Vectors
1	1	82 and *Union of S2-S5, *2,3,4,6,7,8,9,10,11,17,20,21,22,23,25,28,30,31,32,33,37,38,42,43,44,45,46,47,48,54,56,57,58,60,62,63,66,67,69,71,73,75,76,77,78,79,80,81,82,83,84,86,88,90,91,93,94,95,96,97,98
1	2	2,3,4,6,7,8,9,10,11,17,20,21,23,25,28,30,31,32,33,37,38,42,43,44,45,46,47,48,54,56,57,58,60,62,63,66,67,69,71,73,75,76,77,78,79,80,81,82,83,84,86,88,90,91,93,94,95,96,97,98
1	3	2,3,4,6,7,8,9,10,11,17,20,21,23,25,28,30,31,32,33,37,43,44,45,46,47,54,56,57,58,60,63,66,73,76,77,78,79,80,81,82,84,86,88,90,91,94,95,96,97,98
1	4	7,9,17,20,23,31,46,66,84,91,97
1	5	7,9,17,20,22,23,31,46,66,84,91,97
2	1	1,4,5,7,9,10,14,15,17,18,19,21,22,23,24,25,26,30,34,35,37,38,39,41,42,43,44,45,46,47,50,54,56,57,58,61,62,63,64,65,67,68,69,70,71,74,76,77,80,82,83,84,85,86,87,89,90,91,94,96,97,98,99,100
2	2	1,4,5,7,9,10,14,15,17,18,19,21,22,23,24,25,26,30,34,35,37,38,39,41,42,43,44,45,46,47,50,54,56,57,58,61,62,63,64,65,67,68,69,70,71,74,76,77,80,82,83,84,85,86,87,89,90,91,94,96,97,98,99,100
2	3	1,4,5,7,9,10,14,15,17,18,19,21,22,23,24,25,26,30,34,35,37,39,41,42,43,44,45,46,47,50,57,58,61,62,63,64,65,67,68,69,70,71,76,77,82,83,84,87,89,90,94,97,98,99,100
2	4	1,9,10,22,24,26,63,64,77,83,84,89,99,100
2	5	1,9,10,22,24,26,63,64,83,84,89,99,100
3	1	1,2,5,7,9,13,14,15,16,17,21,22,23,24,26,28,29,30,31,32,33,34,35,39,40,43,45,46,48,49,50,51,52,53,54,55,57,60,63,64,66,67,69,70,72,74,75,77,80,81,82,83,86,88,90,91,94,96,97,98,100
3	2	1,2,5,7,9,13,14,15,16,17,21,22,23,24,26,28,29,30,31,32,33,34,35,39,40,43,45,46,48,49,50,51,52,53,54,55,57,60,63,64,66,67,69,70,72,74,75,77,80,81,82,83,86,88,90,91,94,96,97,98,100

Replicate	Scenario	Vectors
3	3	1,5,13,14,15,16,17,21,22,23,24,26,28,29,30,31,32,33,34,35, 39,40,43,45,46,48,49,50,51,52,53,54,55,57,60,63,64,66,69, 72,74,75,80,81,82,83,86,88,90,97,100
3	4	22,26,28,31,54,55,57,64,66,72,74,83,88
3	5	22,26,28,31,54,55,57,64,72,74,88

<u>Log files:</u>

NUT_ISO_CRA1_R*x*S*y*V*nnn*_STEP1.LOG

<u>CMS Libraries for Log Files:</u>

CRA1_NUTR1S1 to CRA1_NUTR1S5
CRA1_NUTR2S1 to CRA1_NUTR2S5
CRA1_NUTR3S1 to CRA1_NUTR3S5

<u>Command Procedures:</u>

EVAL_NUT_CRA1_ISO_TI_RUN.COM
EVAL_NUT_CRA1_ISO_R*x*_S*y*_MASTER_STEP1_2.COM

<u>Command Procedure Input Files:</u>

EVAL_NUT_CRA1_ISO_STEP1_2.INP

<u>CMS Library for Command Procedures and Command Procedure Input Files:</u>

CRA1_EVAL

<u>Code Flow:</u> (RUN PANEL BEFORE RUNNING NUTS ISO and TI)

8.2.3.1.Step 1_2 Codes:

The following codes are run using the EVAL_NUT_CRA1_ISO_STEP1_2.INP command procedure input file after PANEL CON in order to provide the source term input.

NUT

<u>Input files:</u>

NUT_CRA1_ISO_R*x*_S*y*.INP	(CRA1_NUT)
BF2_CRA1_R*x*_S*y*_V*nnn*.INP	(CRA1_BFR*x*S*y*)
BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
PANEL_CON_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_PANEL)

Output files:

NUT_CRA1_ISO_R*x*_S*y*_V*nnn*.CDB (CRA1_NUTR*x*S*y*)
NUT_CRA1_ISO_R*x*_S*y*_V*nnn*.OUT

ALGEBRACDB

Input files:

ALG_NUT_CRA1_POST_S*y*.INP (CRA1_ALG)
NUT_CRA1_ISO_R*x*_S*y*_V*nnn*.CDB

Output files:

ALG_NUT_CRA1_POST_ISO_R*x*_S*y*_V*nnn*.CDB (CRA1_ALG)
ALG_NUT_CRA1_POST_INT*i*_R*x*_S*y*_V*nnn*.CDB (CRA1_ALG)
ALG_NUT_CRA1_POST_ISO_R*x*_S*y*_V*nnn*.DBG

8.2.4. NUTS (Non-Screening (TI))

The NUTS time intrusion runs use a numeric value to associate intrusion time with corresponding input and output files. Each numeric value is dependent on the scenario number. Scenarios 2 and 4 have one intrusion time of 100 years. Scenarios 3 and 5 have four intrusion times: 3000, 5000, 7000, and 9000 years. Table 8-3 lists each intrusion time with the INT*i* designator, where i may be one of the values shown.

Table 8-3 Intrusion Times

File Name Designator *(i)*	Intrusion Time Represented	Applicable Scenarios
1	100 years	2 and 4
3	3000 years	3 and 5
5	5000 years	3 and 5
7	7000 years	3 and 5
9	9000 years	3 and 5

NUTS time intrusions are run once for each replicate/scenario/vector/time intrusion combination. The total of number vectors by scenario will not be known until the runs are made.

Log files:

NUT_CRA1_INT*i*_R*x*S*y*V*nnn*_STEP1_2.LOG

CMS Libraries for Log Files:

CRA1_NUTR1S2 to CRA1_NUTR1S5
CRA1_NUTR2S2 to CRA1_NUTR2S5
CRA1_NUTR3S2 to CRA1_NUTR3S5

Command Procedures:

EVAL_NUT_CRA1_ISO_TI_RUN.COM
EVAL_NUT_CRA1_TI_R*x*_S*y*_MASTER_STEP1_2.COM

Command Procedure Input Files:

EVAL_NUT_CRA1_INT1_STEP1_2.INP
EVAL_NUT_CRA1_INT3_STEP1_2.INP
EVAL_NUT_CRA1_INT5_STEP1_2.INP
EVAL_NUT_CRA1_INT7_STEP1_2.INP
EVAL_NUT_CRA1_INT9_STEP1_2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.2.4.1.Step 1_2 Codes:

The following codes are run using THE EVAL_NUT_CRA1_INT1_STEP1_2.INP, EVAL_NUT_CRA1_INT3_STEP1_2.INP, EVAL_NUT_CRA1_INT5_STEP1_2.INP, EVAL_NUT_CRA1_INT7_STEP1_2.INP, and EVAL_NUT_CRA1_INT9_STEP1_2.INP command procedure input files.

NUT

Input files:

NUT_CRA1_INT*i*_R*x*_S*y*.INP	(CRA1_NUT)
BF2_CRA1_R*x*_S*y*_V*nnn*.INP	(CRA1_BFR*x*S*y*)
BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
PANEL_CON_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_PANEL)
NUT_CRA1_ISO_R*x*_S*y*_V*nnn*.CDB	(CRA1_NUTR*x*S*y*)

Output files:

NUT_CRA1_INT*i*_R*x*_S*y*_V*nnn*.CDB (CRA1_NUTR*x*S*y*)
NUT_CRA1_INT*i*_R*x*_S*y*_V*nnn*.OUT

ALGEBRACDB

Input files:

ALG_NUT_CRA1_POST_S*y*.INP (CRA1_ALG)
NUT_CRA1_INT*i*_R*x*_S*y*_V*nnn*.CDB

Output files:

ALG_NUT_CRA1_POST_INT*i*_R*x*_S*y*_V*nnn*.CDB (CRA1_ALG)
ALG_NUT_CRA1_POST_INT*i*_R*x*_S*y*_V*nnn*.DBG

8.3. PANEL

There are two types of PANEL runs; Concentration (CON), and Time Intrusion (TI) PANEL (REG) is no longer run as TI2000 is included with the TI runs. PANEL (Step 1 and Step 2) to PANEL CON (Step 3) must be run first and the results used between NUTS SCN and NUTS ISO/TI. Subsequently, PANEL TI follows PANEL CON. PANEL TI runs have no impact on NUTS TI.

8.3.1. PANEL

The PANEL flow contains three distinct steps; step 1 includes GENMESH, MATSET, PRELHS, LHS and POSTLHS. Step 2 runs only ALGEBRA. A third step is run for PANEL CON followed by a similar Step 3 for PANEL TI.

Step 1 is run once per replicate. Step 2 is run 100 times per replicate/scenario combination for a total of 300 runs. The step number is embedded in the log file name to easily distinguish between step 1, 2 and 3 runs.

The PANEL includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5, 6

Database Used:

Database: PARAMETER_PROD
The database is used by MATSET and PRELHS.

Log files:

PANEL_CRA1_R*x*_STEP1.LOG
PANEL_ALG_CRA1_R*x*S*y*V*nnn*.LOG

CMS Libraries for Log Files:

CRA1_PANEL

Command Procedures:

EVAL_PANEL_CRA1_STEP1_2_RUN.COM
EVAL_PANEL_CRA1_STEP3_RUN.COM
EVAL_PANEL_CRA1_RUN_MASTER_STEP1_2.COM
EVAL_PANEL_CRA1_RUN_MASTER_STEP3.COM

Command Procedure Input Files:

EVAL_PANEL_CRA1_STEP1.INP
EVAL_PANEL_CRA1_STEP2.INP
EVAL_PANEL_CRA1_CON_STEP3.INP
EVAL_PANEL_CRA1_TI_STEP3.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.3.1.1.Step 1 Code:

Step 1 code is run using the EVAL_PANEL_CRA1_STEP1.INP command procedure input file.

Part of step 1 - run once.

GENMESH

Input files:	
GM_PANEL_CRA1.INP	(CRA1_GM)
Output files:	
GM_PANEL_CRA1_R*x*.CDB	(CRA1_GM)
GM_PANEL_CRA1.DBG	

MATSET

Input files:	
MS_PANEL_CRA1.INP	(CRA1_MS)
GM_PANEL_CRA1_R*x*.CDB	(CRA1_GM)
Output files:	
MS_PANEL_CRA1_R*x*.CDB	(CRA1_MS)
MS_PANEL_CRA1.DBG	

PRELHS

Input files:	
LHS1_PANEL_CRA1_A*x*.INP	(CRA1_LHS)

Output files:
LHS1_PANEL_CRA1_TRN_A*x*.OUT
LHS1_PANEL_CRA1_A*x*.OUT

LHS

Input files:
LHS1_PANEL_CRA1_TRN_A*x*.OUT

Output files:
LHS2_PANEL_CRA1_TRN_A*x*.OUT (CRA1_LHS)
LHS2_PANEL_CRA1_DBG.OUT

POSTLHS

Input files:
MS_PANEL_CRA1_R*x*.CDB (CRA1_MS)
LHS2_PANEL_CRA1_TRN_A*x*.OUT (CRA1_LHS)
LHS3_BF_CRA1.INP (CRA1_LHS)

Output files:
LHS3_PANEL_CRA1_A*x*_R*nnn*.CDB (CRA1_LHS)
LHS3_PANEL_CRA1.DBG
LHS3_PANEL_CRA11.SCR
LHS3_PANEL_CRA12.SCR

8.3.1.2.Step 2 Codes:

Step 2 codes are run using the EVAL_PANEL_CRA1_STEP2.INP command procedure input file.

ALGEBRA

Step 2 – run 100 times (vectors 1-100) per replicate for a total of 300.

Input files:
ALG_PANEL_CRA1.INP (CRA1_ALG)
LHS3_PANEL_CRA1_A*x*_R*nnn*.CDB (CRA1_LHS)

Output files:
ALG_PANEL_CRA1_R*x*_V*nnn*.CDB (CRA1_ALG)
ALG_PANEL_CRA1_R*x*_V*nnn*.DBG

8.3.1.3.Step 3 Codes:

8.3.2. PANEL CON (Concentration)

PANEL Concentration is run for each replicate/scenario/vector combination for a total of 900 runs.

The PANEL Concentration includes the following replicate and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Log files:

PANEL_CON_CRA1_RxSyV*nnn*.LOG

CMS Libraries for Log Files:

CRA1_PANEL

Command Procedures:

EVAL_PANEL_CRA1_STEP3_RUN.COM
EVAL_PANEL_CRA1_RUN_MASTER_STEP3.COM

Command Procedure Input Files:

EVAL_PANEL_CRA1_CON_STEP3.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.3.2.1.Single Step Codes:

The following codes are run using the *EVAL_PANEL_CRA1_CON_STEP3.INP* command procedure input file.

PANEL

Run 100 times (vectors 1-100) per replicate/scenario.

Input files:

ALG_PANEL_CRA1_R*x*_V*nnn*.CDB (CRA1_ALG)

Output files:

PANEL_CON_CRA1_R*x*_S*y*_V*nnn*.CDB (CRA1_PANEL)
PANEL_CON_CRA1_R*x*_S*y*_V*nnn*.DBG

8.3.3. PANEL (Regular)

The PANEL Regular flow is now combined into PANEL TI.

8.3.4. PANEL TI (Time Intrusion)

PANEL TI runs include seven time intrusions for each replicate/scenario/vector combination for a total of 2100 runs.

The PANEL TI includes the following replicates/scenario:

Replicates: 1, 2, 3
Scenarios: 6

The PANEL Time Intrusion runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *tttt* designator, where *tttt* may be one of the following values:

100, 350, 1000, 2000*, 4000, 6000, 9000. *Time 2000 added from PANEL REG.

Log files:

PANEL_TI_CRA1_RxS6V*nnn*_T*tttt*.LOG

CMS Libraries for Log Files:

CRA1_PANEL

Command Procedures:

EVAL_PANEL_CRA1_STEP3_RUN.COM
EVAL_PANEL_CRA1_RUN_MASTER_STEP3.COM

Command Procedure Input Files:

EVAL_PANEL_CRA1_TI_STEP3.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.3.4.1.Single Step Codes:

The following codes are run using the *EVAL_PANEL_CRA1_TI_STEP3.INP* command procedure input file.

PANEL

Run 100 times (vectors 1-100) per replicate/scenario combination.

Input files:

ALG_PANEL_CRA1_R*x*_V*nnn*.CDB	(CRA1_ALG)
ALG2_CRA1_R*x*_S6_V*nnn*.CDB	(CRA1_BFR*x*S6)

Output files:

PANEL_CRA1_R*x*_S6_T*ttttt*_V*nnn*.CDB	(CRA1_PANEL)
PANEL_CRA1_R*x*_S6_T*ttttt*_V*nnn*.DBG	

8.4. DRSPALL

The DRSPALL CRA1 flow contains two distinct steps, step 1 includes GENMESH, MATSET, PRELHS, LHS, and POSTLHS. Step 2 includes DRSPALL.

Step 1 is run once per replicate. Step 2 is run 50 times per replicate/scenario combination for a total of 200 runs. An exception step was added to handle any exceptions (modified vector inputs) and was run in a single step 2.

The DRSPALL CRA1 includes the following replicates and scenarios:

Replicates: 1
Scenarios: 1, 2, 3, 4

Database Used:

Database: PARAMETER_PROD
The database is used by MATSET and PRELHS.

Log files:

DRS_CRA1_R*x*_S*y*_STEP1.LOG
DRS_CRA1_R*x*_S*y*_V*nnnn*.LOG

CMS Libraries for Log Files:

CRA1_DRSR1S1 to CRA1_DRSR1S4

Command Procedures:

EVAL_DRS_RUN.COM
EVAL_DRS_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_DRS_CRA1_STEP1.INP
EVAL_DRS_CRA1_STEP2.INP
EVAL_DRS_CYL_CAV_CRA1_STEP2.INP (Exception runs)
EVAL_DRS_MAXT_CRA1_STEP2.INP (Exception runs)

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.4.1. Step 1 Codes:

Step 1 codes are run using the EVAL_DRS_CRA1_STEP1.INP command procedure input file.

GENMESH

Part of step 1 - run once per replicate.

Input files:

GM_DRS_CRA1_R1.INP	(CRA1_DRS)

Output files:

GM_DRS_CRA1_R1.CDB	(CRA1_DRS)
GM_DRS_CRA1_R1.DBG	

MATSET

Part of step 1- run once per replicate.

Input files:

MS_DRS_CRA1_R1.INP	(CRA1_DRS)
GM_DRS_CRA1_R1.CDB	(CRA1_DRS)

Output files:

MS_DRS_CRA1_R1.CDB	(CRA1_DRS)
MS_DRS_CRA1_R1.DBG	

PRELHS

Part of step 1- run once per replicate.

Input files:

LHS1_DRS_CRA1_A1.INP	(CRA1_DRS)

Output files:

LHS1_DRS_CRA1_TRN_A1.OUT	(CRA1_DRS)
LHS1_DRS_CRA1_A1.OUT	(CRA1_DRS)

LHS

Part of step 1- run once per replicate.

Input files:

LHS1_DRS_CRA1_TRN_A1.OUT	(CRA1_DRS)

Output files:

LHS2_DRS_CRA1_TRN_A1.OUT	(CRA1_DRS)
LHS_DRS2_CRA1_DBG_A1.OUT	(CRA1_DRS)

POSTLHS

Part of step 1- run once per replicate.

Input files:

LHS3_DRS_CRA1_R1.INP	(CRA1_DRS)
LHS2_DRS_CRA1_TRN_A1.OUT	(CRA1_DRS)
MS_DRS_CRA1_R1.CDB	(CRA1_DRS)

Output files:

LHS3_DRS_CRA1_A1_R*nnn*.CDB	(CRA1_DRS)
LHS3_DRS_CRA1.DBG	
LHS3_DRS_CRA11.SCR	
LHS3_DRS_CRA12.SCR	

8.4.2. Step 2 Code:

Step 2 codes are run using the EVAL_DRS_CRA1_STEP2.INP command procedure input file.

DRSPALL

Part of step 2 - run 50 times per replicate/scenario combination.

Input files:

DRS_CRA1_R1_S*y*.DRS	(CRA1_DRS)
LHS3_DRS_CRA1_A1_R*nnn*.CDB	(CRA1_DRS)

Output files:

DRS_CRA1_R1_S*y*_V*nnn*.DBG	
DRS_CRA1_R1_S*y*_V*nnn*.CDB	(CRA1_DRS)

8.4.3. Step 2 Exception Runs:

DRSPALL

Input files:

DRS_CRA1_R1_S2_MAXT.DRS	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S3_V002.DRS	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S4_V002.DRS	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S3_V030.DRS	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S4_V030.DRS	(CRA1_DRS)
LHS3_DRS_CRA1_A1_R*nnn*.CDB	(CRA1_DRS)

Output files:

DRS_CRA1_R1_S2_V030.DBG	
DRS_CRA1_R1_S2_V030.CDB	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S3_V002.DBG	
DRS_CYL_CAV_CRA1_R1_S3_V002.CDB	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S4_V002.DBG	
DRS_CYL_CAV_CRA1_R1_S4_V002.CDB	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S3_V030.DBG	
DRS_CYL_CAV_CRA1_R1_S3_V030.CDB	(CRA1_DRS)
DRS_CYL_CAV_CRA1_R1_S4_V030.DBG	
DRS_CYL_CAV_CRA1_R1_S4_V030.CDB	(CRA1_DRS)

8.5. CUTTINGS_S (CUSP)

The CUTTINGS_S CRA1 flow contains two distinct steps; step 1 includes GENMESH, MATSET, PRELHS, LHS, POSTLHS, and CUTTINGS_S in a pre-processor mode. Step 2 includes CUTTINGS_S in a "regular" mode.

Step 1 is run once per replicate. Step 2 is run 100 times per replicate/scenario/cavity/time intrusion combination for a total of 7,800 runs. The step number is embedded in the log file name to easily distinguish between step 1 and step 2 runs.

Another manual step may be run between steps 1 and 2 depending on Run Control utilization of BATCH queues - a fetch of input files is needed by step 2 runs. The fetch step (step 1A) eliminates the fetching of input files due to time and disk space constraints. All files are fetched from CMS to a directory protected by Access Control Lists that prevent unauthorized access.

The CUTTINGS_S runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

Another designator is used to identify the cavity location. The cavity location is represented in a file name with the *c* designator, where *c* may be one of the following values:

U - upper cavity
M – middle cavity (new for CRA1)
L - lower cavity

The CUTTINGS_S CRA1 includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Database Used:

Database: PARAMETER_PROD
The database is used by MATSET and PRELHS.

Log files:

CUSP_CRA1_Rx_STEP1.LOG
CUSP_CRA1_RxSyV*nnnc*T*ttttt*_STEP2.LOG

CMS Libraries for Log Files:

CRA1_CUSPR1S1 to CRA1_CUSPR1S5
CRA1_CUSPR2S1 to CRA1_CUSPR2S5
CRA1_CUSPR3S1 to CRA1_CUSPR3S5

Command Procedures:

EVAL_CUSP_CRA1_RUN.COM
EVAL_CUSP_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_CUSP_CRA1_STEP1.INP
EVAL_CUSP_CRA1_STEP2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.5.1. Step 1 Codes:

Step 1 codes are run using the EVAL_CUSP_CRA1_STEP1.INP command procedure input file.

GENMESH

Part of step 1 - run once per replicate.

Input files:
GM_CUSP_CRA1.INP (CRA1_GM)

Output files:
GM_CUSP_CRA1.CDB
GM_CUSP_CRA1.DBG

MATSET

Part of step 1- run once per replicate.

<u>Input files:</u>

MS_CUSP_CRA1.INP (CRA1_MS)
GM_CUSP_CRA1.CDB

<u>Output files:</u>

MS_CUSP_CRA1.CDB
MS_CUSP_CRA1.DBG

PRELHS

Part of step 1- run once per replicate.

<u>Input files:</u>

LHS1_CRA1_A*x*.INP (CRA1_LHS)

<u>Output files:</u>

LHS1_CRA1_TRN_A*x*.OUT (CRA1_LHS)
LHS1_CRA1_A*x*.OUT

LHS

Part of step 1- run once per replicate.

<u>Input files:</u>

LHS1_CRA1_TRN_A*x*.OUT (CRA1_LHS)

<u>Output files:</u>

LHS2_CRA1_TRN_A*x*.OUT (CRA1_LHS)
LHS2_CRA1_DBG.OUT

POSTLHS

Part of step 1- run once per replicate.

Input files:

LHS3_CRA1.INP	(CRA1_LHS)
LHS2_CRA1_TRN_A*x*.OUT	
MS_CUSP_CRA1.CDB	

Output files:

LHS3_CRA1_A*x*_R*nnn*.CDB	(CRA1_LHS)
LHS3_CRA1.DBG	

CUTTINGS_S - preprocessor

Part of step 1- run once per replicate.

Input files:

CUSP_CRA1_S1_U_T100.INP	(CRA1_CUSP)
CUSP_CRA1_TEMPLATE.INP	(CRA1_CUSP)
CUSP_CRA1.SDB*	(CRA1_CUSP)
LHS3_CRA1_A*x*_R001.CDB	(CRA1_LHS)

*CUTTINGS requires the file CUSP_CRA1.SDB. For the CRA1 analysis, the file was created using the SQL script CUSP_CRA1.SQL. This script contains two variables that should be changed before it is run, the analysisAbbreviation and effectiveDate. The analysisAbbreviation will be updated with the current analysis and the effectiveDate is the current date.

Output files:

CUSP_CRA1_SDB.ASC	(CRA1_CUSP)
CUSP_CRA1_PRE.DBG	
CUSP_CRA1_SDB.BIN	

CMS Fetch – The below files must reside on the nodes used in the 23,400 runs made to complete CUSP.

Input files:

CUSP_CRA1_S*y*_*c*_T*tttt*.INP	(CRA1_CUSP)
LHS3_CRA1_A*x*_R001.CDB	(CRA1_LHS)
through	
LHS3_CRA1_A*x*_R100.CDB	
CUSP_CRA1_SDB.ASC	(CRA1_CUSP)

Output location:

PAWORK:[NON_HSM.CCA.WORK.CUSP.DATA.INPUTS]

8.5.2. Step 2 Code:

Step 2 codes are run using the EVAL_CUSP_CRA1_STEP2.INP command procedure input file.

CUTTINGS_S

Part of step 2 - run 100 times (vectors 1-100) per replicate/scenario/cavity/time intrusion combination. The number of runs per replicate and scenario follows:

Run 1800 times per replicate for Scenario 1.
Run 1500 times each per replicate for Scenario 2 through 5

Input files:

CUSP_CRA1_S*y*_*c*_T*ttttt*.INP	(CRA1_CUSP)
CUSP_CRA1_SDB.ASC	(CRA1_CUSP)
LHS3_CRA1_A*x*_R*nnn*.CDB	(CRA1_LHS)
BF3_CRA1_R*x*_S*y*_V*nnn*.CDB	(CRA1_BFR*x*S*y*)
SUM_DRS_SPLVOL2.TBL	(CRA1_DRS)

Output files:

CUSP_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.CDB	(CRA1_CUSPR*x*S*y*)
CUSP_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.DBG	

8.6. DBR

The BRAGFLO Direct Brine Release (DBR) CRA1 flow contains three distinct steps; step 1 includes GENMESH and MATSET. Step 2 includes ALGEBRACDB, RELATE, and ICSET. Step 3 includes PREBRAGFLO, BLOWOUT, and POSTBRAGFLO.

Step 1 is run once. Steps 2 and 3 are combined to run 100 times per replicate/scenario/cavity/time intrusion combination for a total of 7,800 runs. The step number is embedded in the log file name to easily distinguish between step 1 and step 2 runs.

The BRAGFLO DBR runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

Another designator is used to identify the cavity location. The cavity location is represented in a file name with the *c* designator, where *c* may be one of the following values:

U - upper cavity
M - upper cavity (new for CRA1)
L - lower cavity

The BRAGFLO DBR CRA1 includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Database Used:

Database: PARMETER_PROD
The database was used by MATSET.

Log files:

DBR_CRA1_STEP1.LOG
DBR_CRA1_R*x*S*y*V*nnnc*T*ttttt*_STEP2_3.LOG

CMS Libraries for Log Files:

CRA1_BFR1S1 to CRA1_BFR3S1 - (step 1 log file)
CRA1_BFR1S1 to CRA1_BFR1S5
CRA1_BFR2S1 to CRA1_BFR2S5
CRA1_BFR3S1 to CRA1_BFR3S5

Command Procedures:

EVAL_DBR_CRA1_RUN.COM
EVAL_DBR_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_DBR_CRA1_STEP1.INP
EVAL_DBR_CRA1_STEP2_3.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.6.1. Step 1 Codes:

Step 1 codes are run using the EVAL_DBR_CRA1_STEP1.INP command procedure input file.

GENMESH

Part of step 1 - run once.

Input files:
GM_DBR_CRA1_DIR_REL.INP (CRA1_GM)

Output files:
GM_DBR_CRA1_PRECAM_DIR_REL.CDB
GM_DBR_CRA1.DBG

MATSET

Part of step 1 - run once.

Input files:

MS_DBR_CRA1_DIR_REL.INP	(CRA1_MS)
GM_DBR_CRA1_PRECAM_DIR_REL.CDB	

Output files:

MS_DBR_CRA1_DIR_REL.CDB	(CRA1_MS)
MS_DBR_CRA1_DIR_REL.DBG	

CMS Fetch – The below files must reside on the nodes used in the runs made to complete DBR.

Input files:

ALG_DBR_CRA1_PRECUSP_DIR_REL.INP	(CRA1_ALG)
ALG_DBR_CRA1_PRE_DIR_REL_S1.INP	(CRA1_ALG)
ALG_DBR_CRA1_PRE_DIR_REL_S2.INP	(CRA1_ALG)
ALG_DBR_CRA1_PRE_DIR_REL_S3.INP	(CRA1_ALG)
ALG_DBR_CRA1_PRE_DIR_REL_S4.INP	(CRA1_ALG)
ALG_DBR_CRA1_PRE_DIR_REL_S5.INP	(CRA1_ALG)
REL_DBR_CUSP_CRA1_DIR_REL.INP	(CRA1_REL)
REL_DBR_BRAG_CRA1_DIR_REL_S1.INP	(CRA1_REL)
REL_DBR_BRAG_CRA1_DIR_REL_S2.INP	(CRA1_REL)
REL_DBR_BRAG_CRA1_DIR_REL_S3.INP	(CRA1_REL)
REL_DBR_BRAG_CRA1_DIR_REL_S4.INP	(CRA1_REL)
REL_DBR_BRAG_CRA1_DIR_REL_S5.INP	(CRA1_REL)
IC_DBR_CRA1_DIR_REL_S1.INP	(CRA1_IC)
IC_DBR_CRA1_DIR_REL_S2.INP	(CRA1_IC)
IC_DBR_CRA1_DIR_REL_S3.INP	(CRA1_IC)
IC_DBR_CRA1_DIR_REL_S4.INP	(CRA1_IC)
IC_DBR_CRA1_DIR_REL_S5.INP	(CRA1_IC)
DBR_BF1_CRA1_DIR_REL_S1_*c*.INP	(CRA1_BF)
DBR_BF1_CRA1_DIR_REL_S2_*c*.INP	(CRA1_BF)
DBR_BF1_CRA1_DIR_REL_S3_*c*.INP	(CRA1_BF)
DBR_BF1_CRA1_DIR_REL_S4_*c*.INP	(CRA1_BF)
DBR_BF1_CRA1_DIR_REL_S5_*c*.INP	(CRA1_BF)
MS_DBR_CRA1_DIR_REL.CDB	(CRA1_MS)

Output location:

PAWORK:[NON_HSM.CCA.WORK.BF.DATA.INPUTS]

8.6.2. Step 2 Codes:

Step 2 codes are run using the EVAL_DBR_CRA1_STEP2_3.INP command procedure input file.

For the CRA1, Step 2 and Step 3 codes are run at the same time for each combination of replicate, scenario, vector, time intrusion, and cavity. The number of runs per combination follows:

Run 1800 times per replicate for Scenario 1.
Run 1500 times each per replicate for Scenario 2 to Scenario 5

ALGEBRACDB

Part of step 2.

<u>Input files:</u>

ALG_DBR_CRA1_PRECUSP_DIR_REL.INP	(CRA1_ALG)
CUSP_CRA1_R*x*_S*y*_V*nnn*_L_T*tttt*.CDB	(CRA1_CUSPR*x*S*y*)

<u>Output files:</u>

DBR_CRA1_R*x*_S*y*_V*nnn*_T*tttt*.CDB
DBR_CRA1_R*x*_S*y*_V*nnn*_T*tttt*.DBG

RELATE_1

Part of step 2.

<u>Input files:</u>

REL_DBR_CUSP_CRA1_DIR_REL.INP	(CRA1_REL)
MS_DBR_CRA1_DIR_REL.CDB	(CRA1_MS)
DBR_CRA1_R*x*S*y*_V*nnn*_T*tttt*.CDB	

<u>Output files:</u>

REL_1_DBR_CUSP_CRA1_R*x*S*y*_V*nnn*_T*tttt*.CDB
REL_1_DBR_CUSP_CRA1_R*x*S*y*_V*nnn*_T*tttt*.DBG

RELATE_2

Part of step 2.

Input files:

REL_1_DBR_CUSP_CRA1_R*x*S*y*_V*nnnn*_T*ttttt*.CDB
BF3_CRA1_R*x*_S*y*_V*nnn*.CDB
REL_DBR_BRAG_CRA1_DIR_REL_S*y*.INP

Output files:

REL_2_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.CDB
REL_2_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.DBG

ICSET

Part of step 2.

Input files:

IC_DBR_CRA1_DIR_REL_S*y*.INP (CRA1_IC)
REL_2_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.CDB

Output files:

IC_DBR_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.CDB
IC_DBR_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.DBG

ALGEBRACDB

Part of step 2.

Input files:

ALG_DBR_CRA1_PRE_DIR_REL_S*y*.INP (CRA1_ALG)
IC_DBR_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.CDB

Output files:

ALG_2_DBR_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.CDB
ALG_2_DBR_CRA1_R*x*S*y*_V*nnn*_T*ttttt*.DBG

8.6.3. Step 3 Codes:

Step 3 codes are run using the EVAL_DBR_CRA1_STEP2_3.INP command procedure input file.

For the CRA1, Step 2 and Step 3 codes are run at the same time for each combination of replicate, scenario, vector, time intrusion, and cavity. The number of runs per combination follows:

Run 1200 times per replicate for Scenario 1.
Run 1000 times per replicate for Scenario 2 to Scenario 5.

PREBRAGFLO

Part of step3.

<u>Input files:</u>

DBR_BF1_CRA1_DIR_REL_Sy_*c*.INP (CRA1_BF)
ALG_2_DBR_CRA1_Rx_Sy_Vnnn_Tttttt.CDB

<u>Output files:</u>

DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.INP (CRA1_BFR*x*S*y*)
BF1_DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.DBG

BRAGFLO

Part of step 3.

<u>Input files:</u>

DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.INP

<u>Output files:</u>

DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.OUT
DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.SUM
DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.BIN
DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.ROT
DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.RIN

POSTBRAGFLO

Part of step 3.

Input files:

ALG_2_DBR_CRA1_R*x*_S*y*_V*nnn*_T*tttt*.CDB
DBR_CRA1_R*x*_S*y*_V*nnn*_*c*_T*tttt*.BIN

Output files:

BF3_DBR_CRA1_R*x*S*y*_V*nnn*_*c*_T*tttt*.CDB (CRA1_BFR*x*S*y*)
BF3_DBR_CRA1_R*x*S*y*_V*nnn*_*c*_T*tttt*.DBG

8.7. SECOTP2D

The SECOTP2D CRA1 contains two distinct steps; step 1 includes GENMESH, MATSET, PRELHS, LHS, POSTLHS, ALGEBRACDB and, RELATE. Step 2 includes PRESECOTP2D, SECOTP2D, and POSTSECOTP2D.

Step 1 is run once per replicate. Step 2 is run 100 times per replicate/mining type combination for a total of 600 runs. The step number is embedded in the log file name to easily distinguish between step 1 and step 2 runs.

The SECOTP2D CRA1 includes the following replicates and vectors:

Replicates: 1, 2, 3
Vectors: 1 - 100
Mining: Full Mining (FM) and Partial Mining (PM)

<u>Database Used:</u>

Database: PARAMETER_PROD
The database view is used by MATSET and PRELHS.

<u>Log files:</u>

ST2D_CRA1_R*x*_STEP1.LOG
ST2D_CRA1_R*x*_V*nnn*_*M*_STEP2.LOG

<u>CMS Libraries for Log Files:</u>

CRA1_ST2D

<u>Command Procedures:</u>

EVAL_ST2D_CRA1_RUN.COM
EVAL_ST2D_CRA1_RUN_MASTER.COM

<u>Command Procedure Input Files:</u>

EVAL_ST2D_CRA1_STEP1.INP
EVAL_ST2D_CRA1_STEP2.INP

<u>CMS Library for Command Procedures and Command Procedure Input Files:</u>

CRA1_EVAL

8.7.1. Step 1 Codes:

Step 1 codes are run using the EVAL_ST2D_CRA1_STEP1.INP command procedure input file.

GENMESH

Part of step 1 - run once per replicate.

Input files:
GM_ST2D_CRA1.INP (CRA1_GM)

Output files:
GM_ST2D_CRA1.CDB
GM_ST2D_CRA1.DBG

MATSET

Part of step 1 - run once per replicate.

Input files:
MS_ST2D_CRA1.INP (CRA1_MS)
GM_ST2D_CRA1.CDB

Output files:
MS_ST2D_CRA1.CDB
MS_ST2D_CRA1.DBG

PRELHS

Part of step 1 - run once per replicate.

Input files:
LHS1_ST2D_CRA1_A*x*.INP (CRA1_LHS)

Output files:
LHS1_ST2D_CRA1_TRN_A*x*.OUT
LHS1_ST2D_CRA1_A*x*.OUT

LHS

Part of step 1 - run once per replicate.

Input files:

LHS1_ST2D_CRA1_TRN_A*x*.OUT

Output files:

LHS2_ST2D_CRA1_TRN_A*x*.OUT (CRA1_LHS)
LHS2_ST2D_CRA1_DBG.OUT

POSTLHS

Part of step 1 - run once per replicate.

Input files:

LHS3_ST2D_CRA1.INP (CRA1_LHS)
LHS2_ST2D_CRA1_TRN_A*x*.OUT
MS_ST2D_CRA1.CDB

Output files:

LHS3_ST2D_CRA1_A*x*_R*nnn*.CDB
LHS3_ST2D_CRA1.DBG
LHS3_ST2D_CRA1_1.SCR
LHS3_ST2D_CRA1_2.SCR

ALGEBRACDB

Part of step 1 - run 100 times (vectors 1-100) per replicate.

Input files:

ALG_ST2D_CRA1.INP (CRA1_ALG)
LHS3_ST2D_CRA1_A*x*_R*nnn*.CDB

Output files:

ALG_ST2D_CRA1_R*x*_V*nnn*.CDB
ALG_ST2D_CRA1_R*x*_V*nnn*.DBG

RELATE

Part of step 1 - run 100 times (vectors 1-100) per replicate.

Input files:

REL_ST2D_CRA1.INP	(CRA1_REL)
GM_ST2D_CRA1.CDB	
ALG_ST2D_CRA1_R*x*_V*nnn*.CDB	

Output files:

REL_ST2D_CRA1_R*x*_V*nnn*.CDB	(CRA1_REL)
REL_ST2D_CRA1_R*x*_V*nnn*.DBG	

8.7.2. Step 2 Codes:

Step 2 codes are run using the EVAL_ST2D_CRA1_STEP2.INP command procedure input file.

PRESECOTP2D

Part of step 2 - run 100 times (vectors 1-100) per replicate/mining type combination.

Input files:

ST2D1_CRA1.INP	(CRA1_ST2D)
MF2K_CRA1_R*x*_F*fff*_*M*.TRN	(CRA1_MF2K)
REL_ST2D_CRA1_R*x*_V*nnn*.CDB	(CRA1_REL)
SUM_ST2D_CRA1_R*x*_TRANSIDX.TBL	(CRA1_ST2D)

Output files:

ST2D2_CRA1_R*x*_V*nnn*_*M*.INP
ST2D1_CRA1_R*x*_V*nnn*_*M*.PRP
ST2D1_CRA1_R*x*_V*nnn*_*M*.VEL
ST2D1_CRA1_R*x*_V*nnn*_*M*.DBG

SECOTP2D

Part of step 2 - run 100 times (vectors 1-100) per replicate/mining type combination.

Input files:

ST2D2_CRA1_R*x*_V*nnn*_*M*.INP

<u>Output files:</u>

ST2D3_CRA1_R*x*_V*nnn*_*M*.BIN
ST2D2_CRA1_R*x*_V*nnn*_*M*.DBG

POSTSECOTP2D

Part of step 2 - run 100 times (vectors 1-100) per replicate/mining type combination.

<u>Input files:</u>

REL_ST2D_CRA1_R*x*_V*nnn*.CDB (CRA1_REL)
ST2D3_CRA1_R*x*_V*nnn*_M.BIN

<u>Output files:</u>

ST2D3_CRA1_R*x*_V*nnn*_*M*.CDB (CRA1_ST2D)
ST2D3_CRA1_R*x*_V*nnn*_*M*.DBG

8.8. SUMMARIZE

Several SUMMARIZE runs are done to provide inputs for CCDF.

8.8.1. SUMMARIZE for BRAGFLO DBR

The BRAGFLO DBR SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *tttt* designator, where *tttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

A cavity designator is also represented in a file name with a *c* where *c* may be one of the following two values:

U	upper cavity
M	middle cavity (new for CRA1)
L	lower cavity

The BRAGFLO DBR SUMMARIZE includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Log files:

SUM_DBR_CRA1R*x*S*y*_T*tttt*.LOG *

* The log file name does not include a cavity designator, so log file names are not unique for a replicate/scenario/time intrusion combination. The CMS library contains a minimum of two generations for a given log file name. For example, if generation 1 of the log file represents the "UP" cavity then generation 2 represents the "LOWER" cavity. Similarly, if generation 1 of the log file represents the "LOWER" cavity then generation 2 represents the "UP" cavity.

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_DBR_CRA1_RUN.COM
EVAL_SUM_DBR_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_DBR_CRA1.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.8.1.1.Single Step Codes:

The following codes are run using the EVAL_SUM_DBR_CRA1.INP command procedure input file.

ALGEBRACDB

Run 100 times (vectors 1-100) per replicate/scenario/cavity/time_intrusion combination.

Input files:

ALG_DBR_CRA1_POST_DIR_REL.INP	(CRA1_ALG)
BF3_DBR_CRA1_R*x*S*y*_V*nnn*_*c*_T*tttt*.CDB	(CRA1_BFR*x*S*y*)

Output files:

DBR_POST_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.CDB
DBR_POST_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.DBG

SUMMARIZE

Run once per replicate/scenario/cavity/time intrusion combination.

Input files:

SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*.INP (CRA1_SUM)
DBR_POST_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.CDB *

* Referenced in SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*.INP files that contain names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the CRA1 Coordinator(s). All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:

SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*.TBL (CRA1_SUM)
SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*.LOG
SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*_ERROR.LOG

8.8.2. SUMMARIZE for CUTTINGS_S

The CUTTINGS_S SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

A cavity designator is also represented in a file name with a *c* where *c* may be one of the following values:

U - upper cavity
M – middle cavity (new for CRA1)
L - lower cavity

The CUTTINGS_S SUMMARIZE includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Log files:

SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.LOG

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_CUSP_RUN.COM
EVAL_SUM_CUSP_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_CUSP_CRA1.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.8.2.1.Single Step Code:

The following codes are run using the EVAL_SUM_CUSP_CRA1.INP command procedure input file.

SUMMARIZE

Run once per replicate/scenario/cavity/time_intrusion combination.

Input files:

SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.INP	(CRA1_SUM)
CUSP_CRA1_R*x*_S*y*_V*nnn*_*c*_T*ttttt*.CDB *	

* Referenced in SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.INP files which contain names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:

SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.TBL (CRA1_SUM)
SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.LOG
SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*_ERROR.LOG

8.8.3. SUMMARIZE for NUTS

The NUTS SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

No time used	Scenario 1
100, 350	Scenarios 2 & 4
1000, 3000, 5000, 7000, 9000	Scenarios 3 &5

The NUTS ISO SUMMARIZE includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2, 3, 4, 5

Log files:

SUM_NUT_CRA1_R*x*S*y*.LOG	(Scenario 1)
SUM_NUT_CRA1_R*x*S*y*_T*ttttt*.LOG	(Scenarios 2-5)

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_NUT_CRA1_RUN.COM
EVAL_SUM_NUT_RUN_CRA1_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_NUT_CRA1_S1.INP
EVAL_SUM_NUT_CRA1_S2_S5.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

The following code was run using the EVAL_SUM_NUT_CRA1_S1.INP command procedure input file.

SUMMARIZE (S1 Only)

Run once per replicate/scenario combination - Scenario 1 only.

Input files:

SUM_NUT_CRA1_R*x*_S1.INP (CRA1_SUM)
ALG_NUT_CRA1_POST_ISO_R*x*_S1_V*nnn*.CDB *

* Referenced in SUM_NUT_CRA1_Rx_S1.INP file that contains names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:

SUM_NUT_CRA1_R*x*_S1.TBL (CRA1_SUM)
SUM_NUT_CRA1_R*x*_S1.LOG
SUM_NUT_CRA1_R*x*_S1_ERROR.LOG

The following code was run using the EVAL_SUM_NUT_CRA1_S2_S5.INP command procedure input file.

SUMMARIZE (S2-S5)

Run once per replicate/scenario/time intrusion combination - Scenarios 2-5.

Input files:

Summarize Input File   NUT Files Referenced*

SUM_NUT_CRA1_R*x*_S2_T100.INP
SUM_NUT_CRA1_R*x*_S2_T350.INP
SUM_NUT_CRA1_R*x*_S3_T1000.INP
SUM_NUT_CRA1_R*x*_S3_T3000.INP
SUM_NUT_CRA1_R*x*_S3_T5000.INP
SUM_NUT_CRA1_R*x*_S3_T7000.INP
SUM_NUT_CRA1_R*x*_S3_T9000.INP
SUM_NUT_CRA1_R*x*_S4_T100.INP
SUM_NUT_CRA1_R*x*_S4_T350.INP
SUM_NUT_CRA1_R*x*_S5_T1000.INP
SUM_NUT_CRA1_R*x*_S5_T3000.INP
SUM_NUT_CRA1_R*x*_S5_T5000.INP
SUM_NUT_CRA1_R*x*_S5_T7000.INP
SUM_NUT_CRA1_R*x*_S5_T9000.INP
ALG_NUT_CRA1_POST_INT1_R*x*_S2_V*nnn*.CDB
ALG_NUT_CRA1_POST_ISO_R*x*_S2_V*nnn*.CDB
ALG_NUT_CRA1_POST_ISO_R*x*_S3_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT3_R*x*_S3_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT5_R*x*_S3_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT7_R*x*_S3_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT9_R*x*_S3_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT1_R*x*_S4_V*nnn*.CDB
ALG_NUT_CRA1_POST_ISO_R*x*_S4_V*nnn*.CDB
ALG_NUT_CRA1_POST_ISO_R*x*_S5_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT3_R*x*_S5_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT5_R*x*_S5_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT7_R*x*_S5_V*nnn*.CDB
ALG_NUT_CRA1_POST_INT9_R*x*_S5_V*nnn*.CDB

* Referenced in SUM_NUT_CRA1_R*x*_S*y*_T*ttttt*.INP files that contain names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:

SUM_NUT_CRA1_R*x*_S*y*_T*ttttt*.TBL (CRA1_SUM)
SUM_NUT_CRA1_R*x*_S*y*_T*ttttt*.LOG
SUM_NUT_CRA1_R*x*_S*y*_T*ttttt*_ERROR.LOG

8.8.4. SUMMARIZE for PANEL (Concentration)

The PANEL Concentration SUMMARIZE includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 1, 2

Log files:

SUM_PANEL_CON_CRA1_R*x*S*y*_STEP2.LOG

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_PANEL_CRA1_RUN.COM
EVAL_SUM_PANEL_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_PANEL_CON_CRA1_STEP2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

The following codes are run using the EVAL_SUM_PANEL_CON_CRA1_STEP2.INP command procedure input file.

8.8.4.1.Single Step Code:

SUMMARIZE

Run once per replicate/scenario combination.

Input files:

SUM_PANEL_CRA1_CON_R*x*_S*y*.INP (CRA1_SUM)
ALG_PANEL_CRA1_CON_R*x*_S*y*_V*nnn*.CDB *

* Referenced in SUM_PANEL_CRA1_CON_R*x*_S*y*.INP files that contain names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:

SUM_PANEL_CRA1_CON_R*x*_S*y*.TBL (CRA1_SUM)
SUM_PANEL_CRA1_CON_R*x*_S*y*.LOG
SUM_PANEL_CRA1_CON_R*x*_S*y*_ERROR.LOG

8.8.5. SUMMARIZE for PANEL (Time Intrusion)

The SUMMARIZE for PANEL (Time Intrusion) includes the following replicates and scenarios:

Replicates: 1, 2, 3
Scenarios: 6

These runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 2,000, 4000, 6000, 9000

Log files:

SUM_PANEL_TI_CRA1_R*x*S6_T*ttttt*_STEP1_2.LOG

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_PANEL_CRA1_RUN.COM
EVAL_SUM_PANEL_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_PANEL_TI_CRA1_STEP1_2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

The following codes are run using the EVAL_SUM_PANEL_TI_CRA1_STEP1_2.INP command procedure input file.

8.8.5.1.Single Step Codes:

SUMMARIZE

Run once per replicate/scenario/time intrusion combination.

Input files:
SUM_PANEL_CRA1_R*x*_S6_*ttttt*.INP (CRA1_SUM)
ALG_PANEL_CRA1_R*x*_S6_V*nnn*_T*ttttt*.CDB *

* Referenced in SUM_PANEL_CRA1_R*x*_S6_*ttttt*.INP files which contain names of input data files used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

Output files:
SUM_PANEL_CRA1_R*x*_S6_*ttttt*.TBL (CRA1_SUM)
SUM_PANEL_CRA1_R*x*_S6_*ttttt*.LOG
SUM_PANEL_CRA1_R*x*_S6_*ttttt*_ERROR.LOG

8.8.6. SUMMARIZE for SECOTP2D

The SUMMARIZE for SECOTP2D includes the following replicates:

Replicates: 1, 2, 3
Full and Partial Mining

These runs use a numeric value to associate mining type with corresponding input and output files. Each mining type is represented in a file name with the M designator, where M may be one of the following values:

PM, FM (PM = partial mining, FM = full mining)

Log files:

SUM_ST2D3_CRA1_R*x*_*M*.LOG

CMS Libraries for Log Files:

CRA1_SUM

Command Procedures:

EVAL_SUM_ST2D_CRA1_RUN.COM
EVAL_SUM_ST2D_CRA1_RUN_MASTER.COM

Command Procedure Input Files:

EVAL_SUM_ST2D_CRA1.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.8.6.1.Single Step Code:

SUMMARIZE

Run once per replicate/mining type combination.

Input files:

SUM_ST2D_CRA1_R*x*_*M*.INP	(CRA1_SUM)
ST2D3_CRA1_R*x*_V*nnn*_*M*.CDB *	(CRA1_ST2D)

* Referenced in SUM_ST2D_CRA1_R*x*_*M*.INP files which contain names of the input data file used by SUMMARIZE.

Files were copied to a reference disk as part of a CRA1 script, or were manually fetched to a reference disk from CMS by the Run Control Coordinator. All reference disks are controlled with Access Control Lists where read-only access is granted to approved users and write-level access is granted only to the Run Control Coordinator.

<u>Output files:</u>

SUM_ST2D3_CRA1_R*x*_*M*.TBL (CRA1_SUM)
SUM_ST2D_CRA1_R*x*_*M*.LOG
SUM_ST2D_CRA1_R*x*_*M*_ERROR.LOG

8.9. CCDFGF

The run for CCDF consists of four separate code runs, EPAUNI, PRECCDFGF, and CCDFGF. CCDFSUM follows as a separate run.

Step1 runs EPAUNI, and Step 2, PRECCDFGF are run once per replicate. Step 3 includes PRELHS, LHS, and CCDFGF in "regular" mode. A fourth step is the running of CCDFSUM and is run once pre replicate. The step number is embedded in the log file name to easily distinguish between step 1 and step 4 runs.

8.9.1. EPAUNI

The EPAUNI includes the following replicates:

Replicates: 1, 2, 3

Log files:

CCGF_CRA1_R*x*_STEP1.LOG

CMS Libraries for Log Files:

CRA1_CCGF

Command Procedures:

EVAL_CCGF_CRA1_RUN.COM

Command Procedure Input Files:

EVAL_CCGF_CRA1_STEP1.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.9.1.1.Step 1 Codes:

Step 1 codes are run using the EVAL_CCGF_CRA1_STEP1.INP command procedure input file.

EPAUNI

Part of step 1 - run once per replicate.

Input files:

EPU_CRA1_RH.INP	(CRA1_EPU)
EPU_CRA1_RH_MISC.INP	(CRA1_EPU)

Output files:

EPU_CRA1_CRH.DAT	(CRA1_EPU)

Input files:

EPU_CRA1_CH.INP	(CRA1_EPU)
EPU_CRA1_CH_MISC.INP	(CRA1_EPU)

Output files:

EPU_CRA1_CCH.DAT	(CRA1_EPU)

8.9.2. PRECCDFGF

PRECCDFGF is a basic fetch routine. Numerous files are required that are produced during the SUMMARIZE runs (*.TBL files) and are fetched for processing during Step 3. The PRECCDFGF includes the following replicates:

Replicates: 1, 2, 3

Log files:

CCGF_CRA1_R*x*_STEP2.LOG

CMS Libraries for Log Files:

CRA1_CCGF

Command Procedures:

EVAL_CCGF_CRA1_RUN.COM

Command Procedure Input Files:

EVAL_CCGF_CRA1_STEP2.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.9.2.1.Step 2 Codes:

Step 2 codes are run using the EVAL_CCGF_CRA1_STEP2.INP command procedure input file.

PRECCDFGF

Step 2 - run once per replicate.

<u>Input files</u>:

CCGF_CRA1_MS.CDB	(CRA1_MS)
EPU_CRA1_RH_DAT.OUT	(CRA1_EPU)
EPU_CRA1_CH_DAT.OUT	(CRA1_EPU)

This CUSP SUMMARIZE file is arbitrarily chosen to get the number of vectors, drill diameter, and cavings diameter.

SUM_CUSP_CRA1_R*x*_S1_L_T5000.TBL	(CRA1_SUM)

The BRAGFLO DBR SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

SUM_DBR_CRA1_R*x*_S*y*_*c*_T*ttttt*.TBL	(CRA1_SUM)

The CUTTINGS_S SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 3000, 5000, 10000	Scenario 1
550, 750, 2000, 4000, 10000	Scenarios 2 & 4
1200, 1400, 3000, 5000, 10000	Scenarios 3 &5

SUM_CUSP_CRA1_R*x*_S*y*_*c*_T*ttttt*.TBL	(CRA1_SUM)
SUM_ST_CRA1_R*x*_S1.TBL	(CRA1_SUM)
SUM_ST_CRA1_R*x*_S2.TBL	(CRA1_SUM)

The NUTS SUMMARIZE runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

No time used	Scenario 1
100, 350	Scenarios 2 & 4
1000, 3000, 5000, 7000, 9000	Scenarios 3 &5

SUM_NUT_CRA1_R*x*_S1.TBL	(CRA1_SUM)
SUM_NUT_CRA1_R*x*_S*y*_T*ttttt*.TBL	(CRA1_SUM)
SUM_PANEL_CRA1_CON_R*x*_S1.TBL	(CRA1_SUM)
SUM_PANEL_CRA1_CON_R*x*_S2.TBL	(CRA1_SUM)

The PANEL S6 runs use a numeric value to associate intrusion time with corresponding input and output files. Each intrusion time is represented in a file name with the *ttttt* designator, where *ttttt* may be one of the following values:

100, 350, 1000, 2000, 4000, 6000, 9000

SUM_PANEL_CRA1_R*x*_S6_*ttttt*.TBL	(CRA1_SUM)
SUM_ST2D3_CRA1_R*x*_PM.TBL	(CRA1_SUM)
SUM_ST2D3_CRA1_R*x*_FM.TBL	(CRA1_SUM)
LHS2_CRA1_DBG_A*x*.OUT	(CRA1_LHS)
INTRUSIONTIMES.IN	(CRA1_CCGF)

<u>Output files:</u>

CCGF_CRA1_RELTAB_R*x*.DAT	(CRA1_CCGF)

8.9.3. CCDFGF

The CCDFGF includes the following replicates:

Replicates: 1, 2, 3

<u>Database Used:</u>

Database: PARAMETER_PROD
The database view was used by PRELHS.

<u>Log files:</u>

CCGF_CRA1_R*x*_STEP3.LOG

<u>CMS Libraries for Log Files:</u>

CRA1_CCGF

Command Procedures:

EVAL_CCGF_CRA1_RUN.COM

Command Procedure Input Files:

EVAL_CCGF_CRA1_STEP3.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.9.3.1.Step 3 Codes:

Step 3 codes are run using the EVAL_CCGF_CRA1_STEP3.INP command procedure input file.

CCDFGF

Part of step3 - run once per replicate.

Input files:

CCGF_CRA1_RELTAB_R*x*.DAT	(CRA1_CCGF)
CCGF_CRA1_CONTROL.INP	(CRA1_CCGF)

Output files:

CCGF_CRA1_R*x*.OUT	(CRA1_CCGF)
CCGF_CRA1_R*x*.PRT	
CCGF_CRA1_CUTTING_R*x*.TRN	
CCGF_CRA1_BLOWOUT_R*x*.TRN	
CCGF_CRA1_SPALLING_R*x*.TRN	
CCGF_CRA1_NUT_R*x*.TRN	
CCGF_CRA1_SECOTP_R*x*.TRN	

8.9.4. CCDFSUM

CCDFSUM will be run once per replicate.

Log files:

CCDFSUM_CRA1_R1.LOG
CCDFSUM_CRA1_R2.LOG
CCDFSUM_CRA1_R3.LOG

CMS Libraries for Log Files:

CRA1_CCGF

Command Procedures:

EVAL_CCDFSUM_CRA1_RUN.COM

Command Procedure Input Files:

EVAL_CCDFSUM_CRA1_R1.INP
EVAL_CCDFSUM_CRA1_R2.INP
EVAL_CCDFSUM_CRA1_R3.INP
EVAL_CCDFSUM_CRA1_ALL.INP

CMS Library for Command Procedures and Command Procedure Input Files:

CRA1_EVAL

8.9.4.1.Step 4 Codes

The following codes are run using the EVAL_CCDFSUM_CRA1_R*x*.INP command procedure input file.

CCDFSUM_PLT_1

Input files:

CCGF_CRA1_R*x*.OUT	(CRA1_CCGF)
CCDFSUM_CRA1_R*x*.INP	(CRA1_CCGF)

Output files:

CCDFSUM_CRA1_1_R*x*.PLT	(CRA1_CCGF)

CCDFSUM_ADOBE_1

Input files:

CCGF_CRA1_R*x*.OUT	(CRA1_CCGF)
CCDFSUM_CRA1_R*x*.INP	(CRA1_CCGF)

Output files:

CCDFSUM_CRA1_1_R*x*.PI	
CCDFSUM_CRA1_1_R*x*.PI_1	(CRA1_CCGF)
through	
CCDFSUM_CRA1_1_R*x*.PI_*nnn*	

The following codes are run using the EVAL_CCDFSUM_CRA1_ALL.INP command procedure input file.

CCDFSUM_PLT_1

Input files:

CCGF_CRA1_R*x*.OUT	(CRA1_CCGF)
CCDFSUM_CRA1_ALL.INP	(CRA1_CCGF)

Output files:

CCDFSUM_CRA1_ALL.PLT	(CRA1_CCGF)

CCDFSUM_ADOBE_1

Input files:

CCGF_CRA1_R*x*.OUT	(CRA1_CCGF)
CCDFSUM_CRA1_ALL.INP	(CRA1_CCGF)

Output files:

CCDFSUM_CRA1_ALL.AI	
CCDFSUM_CRA1_ALL.AI_1	(CRA1_CCGF)
through	
CCDFSUM_CRA1_ALL.AI_435	

APPENDIX A - CUSP_CRA1.SQL

```
----------------------------------------------------------------------------
-- start of script
----------------------------------------------------------------------------

  set nocount on

----------------------------------------------------------------------------
-- declare & initialize relevant arguments
----------------------------------------------------------------------------

  declare @analysisAbbreviation varchar(50)
  declare @effectiveDate varchar(10)
  declare @isForCompliance int
  declare @analysisID int

----------------------------------------------------------------------------
-- these variable need to be changed for each new run
----------------------------------------------------------------------------

  set @analysisAbbreviation = 'CRA1'
  set @effectiveDate = '2003-05-08'

----------------------------------------------------------------------------
-- get the analysis ID and type (for compliance or not)
----------------------------------------------------------------------------

  set @analysisID = 0
  set @isForCompliance = 1

  select
     @analysisID = a.[analysisID]
  ,  @isForCompliance = a.[isForCompliance]
  from
     [dbo].[analysis] a
  where
     a.[abbreviation] = @analysisAbbreviation

  if ( @analysisID = 0 ) begin
     print '* * * error: the analysis ' + @analysisAbbreviation + ' was not found * * *'
     return
  end

----------------------------------------------------------------------------
```

```
-- create a temporary table for the IDs of the parameters eligible for retrieval
--------------------------------------------------------------------------------

  create table
    #RetrievalParameter
  (
    parameterRecordID int not null
  , parameterID int not null
  )

--------------------------------------------------------------------------------
-- create a temporary table for the output
--------------------------------------------------------------------------------

  create table
    #Output
  (
    value char(155)
  )

--------------------------------------------------------------------------------
-- get a list of active parameter records to retrieve that are assigned to the analysis
--------------------------------------------------------------------------------

  insert into
    #RetrievalParameter
  (
    parameterRecordID
  , parameterID
  )
  select
    a.[parameterRecordID]
  , a.[parameterID]
  from
    [dbo].[parameterRecord] a
  inner join
    [dbo].[parameter] b
  on
    a.[parameterID] = b.[parameterID]
  where
    a.[analysisID] = @analysisID
  and
    b.[isActive] = 1
  and
    a.[effectiveDateTime] =
    (
      select
        max( a1.[effectiveDateTime] )
      from
        [dbo].[parameterRecord] a1
      where
        a1.[analysisID] = a.[analysisID]
      and
        a1.[parameterID] = a.[parameterID]
      and
        a1.[isActive] = 1
      and
        a1.[effectiveDateTime] < @effectiveDate
    )

-----------------------------------------------------------------------------------------
-- get a list of active parameter records to retrieve that are not assigned to the analysis
-----------------------------------------------------------------------------------------

  if ( @isForCompliance = 1 ) begin

    insert into
      #RetrievalParameter
    (
      parameterRecordID
    , parameterID
    )
    select
      a.[parameterRecordID]
    , a.[parameterID]
    from
      [dbo].[parameterRecord] a
    inner join
      [dbo].[parameter] b
    on
      a.[parameterID] = b.[parameterID]
    where
      b.[isActive] = 1
  -- * * * added criteria to avoid retrieving PAD values * * *
 --    and
 --      a.analysisID = 1
  -- * * * added criteria to avoid retrieving PAD values * * *
    and
      a.[effectiveDateTime] =
    (
      select
        max( a1.[effectiveDateTime] )
```

```
         from
            [dbo].[parameterRecord] a1
         where
            a1.[parameterID] = a.[parameterID]
         and
            a1.[isActive] = 1
         and
            a1.[qualificationID] > 0
         and
            a1.[effectiveDateTime] < @effectiveDate
      )
      and
         not exists
         (
            select
               a11.[parameterID]
            from
               #RetrievalParameter a11
            where
               b.[parameterID] = a11.[parameterID]
         )

   end

-- the analysis is not for compliance
   else begin

      insert into
         #RetrievalParameter
      (
         parameterRecordID
      ,  parameterID
      )
      select
         a.[parameterRecordID]
      ,  a.[parameterID]
      from
         [dbo].[parameterRecord] a
      inner join
         [dbo].[parameter] b
      on
         a.[parameterID] = b.[parameterID]
      where
         b.[isActive] = 1
      and
         a.[effectiveDateTime] =
      (
         select
            max( a1.[effectiveDateTime] )
         from
            [dbo].[parameterRecord] a1
         where
            a1.[parameterID] = a.[parameterID]
         and
            a1.[isActive] = 1
         and
         a1.[effectiveDateTime] < @effectiveDate
      )
      and
         not exists
         (
            select
               a11.[parameterID]
            from
               #RetrievalParameter a11
            where
               b.[parameterID] = a11.[parameterID]
         )

   end

-----------------------------------------------------------------------------
--  parse out the values for each parameter
-----------------------------------------------------------------------------

   declare @parameterID varchar(50)
   declare @materialAbbreviation varchar(50)
   declare @propertyAbbreviation varchar(50)
   declare @distributionName varchar(50)
   declare @unit varchar(50)
   declare @theXml varchar(8000)

   declare @theMean varchar(16)
   declare @theMedian varchar(16)
   declare @theMinimum varchar(16)
   declare @theMaximum varchar(16)
   declare @theValueTypeValue varchar(16)
   declare @theValue varchar(16)

   declare @outputValue varchar(156)

   declare
```

```
  parameterValue
cursor for
select
  a.[parameterID]
, c.[abbreviation]
, d.[abbreviation]
, upper( f.[name] )
, g.[abbreviation]
, e.[value]
from
  #RetrievalParameter a
inner join
  [dbo].[parameter] b
on
  a.[parameterID] = b.[parameterID]
inner join
  [dbo].[material] c
on
  b.[materialID] = c.[materialID]
inner join
  [dbo].[property] d
on
  b.[propertyID] = d.[propertyID]
inner join
  [dbo].[parameterRecord] e
on
  a.[parameterRecordID] = e.[parameterRecordID]
inner join
  [dbo].[distribution] f
on
  e.[distributionID] = f.[distributionID]
inner join
  [dbo].[unit] g
on
  d.[unitID] = g.[unitID]
order by
  c.[abbreviation]
, d.[abbreviation]

open
  parameterValue
fetch next from
  parameterValue
into
  @parameterID
, @materialAbbreviation
,  @propertyAbbreviation
,  @distributionName
,  @unit
,  @theXml

while ( @@fetch_status = 0 ) begin

   set @theMedian = ”
   set @theMean = ”
   set @theMinimum = ”
   set @theMaximum = ”
   set @theValue = ”
   set @theValueTypeValue = ”

-- get core values
   execute getElementValueFromXml ’mean’, @theXml, @theMean out
   execute getElementValueFromXml ’minimum’, @theXml, @theMinimum out
   execute getElementValueFromXml ’maximum’, @theXml, @theMaximum out

   if  (@distributionName <> ’TRIANGULAR’ ) begin
      execute getElementValueFromXml ’median’, @theXml, @theMedian out
   end
   else begin
      execute getElementValueFromXml ’mode’, @theXml, @theMedian out
   end

   if  (@distributionName = ’CONSTANT’ ) begin
      execute getElementValueFromXml ’value’, @theXml, @theValue out
   end

-- ---------------------------------------------------------------------
-- parse distributions
-- ---------------------------------------------------------------------

   if  (@distributionName = ’CONSTANT’ ) begin

      set @theMean = @theValue
      set @theMedian = @theValue
      set @theMinimum = @theValue
      set @theMaximum = @theValue

   end

-- format output
   if ( cast( @theMean as float ) >= 0 ) begin
      set @theMean = ’ ’+ @theMean
```

```
    end
    if ( cast( @theMedian as float ) >= 0 ) begin
      set @theMedian = ''+ @theMedian
    end
    if ( cast( @theMinimum as float ) >= 0 ) begin
      set @theMinimum = ''+ @theMinimum
    end
    if ( cast( @theMaximum as float ) >= 0 ) begin
      set @theMaximum = ''+ @theMaximum
    end

    set @theValueTypeValue = '0.00000000e+000’
    set @theValue = '0.0000000e+000’

    if  (@distributionName = 'CUMULATIVE’) or ( @distributionName = 'DELTA’)
begin
      execute getNodeSetFromXml 'percentiles’,  @theXml, @theXml out
    end
    else if ( @distributionName = 'STUDENT’) begin
      execute getNodeSetFromXml 'values’,  @theXml, @theXml out
    end

  -- loop through all values
    while (1 = 1) begin

      if  ( @distributionName = 'CUMULATIVE’) or ( @distributionName = 'DELTA’
) begin
        execute getAttributeValueFromXml 'at’, @theXml, @theValueTypeValue out
        execute getElementValueFromXml 'value’, @theXml, @theValue out
        execute removeElementFromXml 'value’, @theXml, @theXml out
      end

      else if ( @distributionName = 'STUDENT’) begin
        set @theValueTypeValue = '0.00000000e+000’
        execute getElementValueFromXml 'value’, @theXml, @theValue out
        execute removeElementFromXml 'value’, @theXml, @theXml out
      end

    -- format output
      if ( cast( @theValueTypeValue as float ) >= 0 ) begin
        set @theValueTypeValue = ''+ @theValueTypeValue
      end
      if ( cast( @theValue as float ) >= 0 ) begin
        set @theValue = ''+ @theValue
      end
    -- create output
       set @outputValue = ''
       set @outputValue = @outputValue + cast( replicate( ' ', 14 - len( @parameterID)
) + @parameterID as char(15) )
       set @outputValue = @outputValue + cast( @materialAbbreviation as char(9) )
       set @outputValue = @outputValue + cast( @propertyAbbreviation as char(9) )
       set @outputValue = @outputValue + cast( @theMean + ' ' as char(16) )
       set @outputValue = @outputValue + cast( @theMedian + ' ' as char(16) )
       set @outputValue = @outputValue + cast( @theMinimum + ' ' as char(16) )
       set @outputValue = @outputValue + cast( @theMaximum + ' ' as char(16) )
       set @outputValue = @outputValue + cast( @theValueTypeValue + ' ' as char(16)
)
       set @outputValue = @outputValue + cast( @theValue + ' ' as char(16) )
       set @outputValue = @outputValue + cast( @distributionName as char(13) )
       set @outputValue = @outputValue + cast( @unit as char(13) )

    -- add row to table
       insert into
          #Output
       (
       value
       )
          values
       (
          @outputValue
       )

    -- constant, uniform, normal, triangular, etc. have only one row of data
       if ( @distributionName <> 'CUMULATIVE' ) and ( @distributionName <>
'DELTA' ) and ( @distributionName <> 'STUDENT' ) begin
          break
       end

    -- see if there is anything more to parse
       if ( len( @theXml ) = 0 ) begin
          break
       end

    end -- while

 -- get next parameter
    fetch next from
       parameterValue
    into
       @parameterID
    ,  @materialAbbreviation
```

```
        ,  @propertyAbbreviation
        ,  @distributionName
        ,  @unit
        ,  @theXml

    end

    close
        parameterValue
    deallocate
        parameterValue

--------------------------------------------------------------------------------
-- retrieve output
--------------------------------------------------------------------------------

    select
        *
    from
        #Output

--------------------------------------------------------------------------------
--  destroy temporary tables
--------------------------------------------------------------------------------

    drop table
        #Output

    drop table
        #RetrievalParameter

--------------------------------------------------------------------------------
--  end of script
--------------------------------------------------------------------------------
```

APPENDIX B – REFERENCES

1. Analysis Plan (AP-105). 2003. "Analysis Plan For Compliance Recertification Application Performance Assessment Calculations." Sandia National Laboratories. Sandia WIPP Central Files ERMS#525252.

2. U.S. DOE (U.S. Department of Energy). 1996. "Title 40 CFR 191 Compliance Certification Application for the Waste Isolation Pilot Plant." DOE/CAO-1996-2184. Carlsbad, NM: U.S. Department of Energy, Waste Isolation Pilot Plant, Carlsbad Area Office.

3. WIPP PA (Performance Assessment). 2003. "Software Configuration Management System (SCMS) Plan." Sandia National Laboratories. Sandia WIPP Central Files ERMS#524707.