ORNL/TM-2015/726

Nuclear Facility Accident (NFAC) Unit Test Report For HPAC Version 6.3

Ronald W. Lee
Robert W. Morris
C. David Sulfredge

December, 2015

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2015/726

Computational Sciences and Engineering Division

NUCLEAR FACILITY ACCIDENT (NFAC) UNIT TEST REPORT FOR HPAC VERSION 6.3

Ronald W. Lee
Robert W. Morris
C. David Sulfredge

Date Published: December, 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6285
managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

ACRONYMS

AGR	Advanced Gas-Cooled Reactor
API	Application Programming Interface
ATR	Advanced Thermal Reactor
BWR	Boiling Water Reactor
DTRA	Defense Threat Reduction Agency
FBR	Fast Breeder Reactor
GCHWR	Gas-Cooled Heavy Water Reactor
GCR	Gas-Cooled Reactor
GUI	Graphical User Interface
HPAC	Hazard Prediction and Assessment Capability
HTGR	High-Temperature Gas Reactor
HWGCR	Heavy Water Gas-Cooled Reactor
HWLWR	Heavy Water Light Water Reactor
LGR	Light-Water Cooled Graphite-Moderated Reactor
LMFBR	Liquid-Metal Fast Breeder Reactor
MELCOR	Methods for Estimation of Leakages and Consequences of Releases
MWt	Megawatt thermal
NFAC	Nuclear Facility Accident Model
ORIGEN	Oak Ridge Isotope Generator
OSGi	Open Service Gateway Initiative
PHWR	Pressurized Heavy Water Reactor
PIR	Percent Inventory Release
RASCAL	Radiological Assessment Systems for Consequence Analysis
PWR	Pressurized Water Reactor
RBMK	(Russian) Reaktor Bolshoy Moschnosti Kanainy
RTH	Radiological Transport for HPAC
SCIPUFF	Second-order Closure Integrated Puff Model
SGHWR	Steam-Generating Heavy Water Reactor
SPCR	Software Problem/Change Report
VVER	(Russian) Water-Water Energy Reactor

1. OVERVIEW

NFAC's responsibility as an HPAC component is three-fold. First, it must present an interactive graphical user interface (GUI) by which users can view and edit the definition of an NFAC incident. Second, for each incident defined, NFAC must interact with RTH to create activity table inputs and associate them with pseudo materials to be transported via SCIPUFF. Third, NFAC must create SCIPUFF releases with the associated pseudo materials for transport and dispersion. The goal of NFAC unit testing is to verify that the inputs it produces are correct for the source term or model definition as specified by the user via the GUI. As shown in Fig. 1, NFAC contains 22 distinct source term models, all of which are tested.

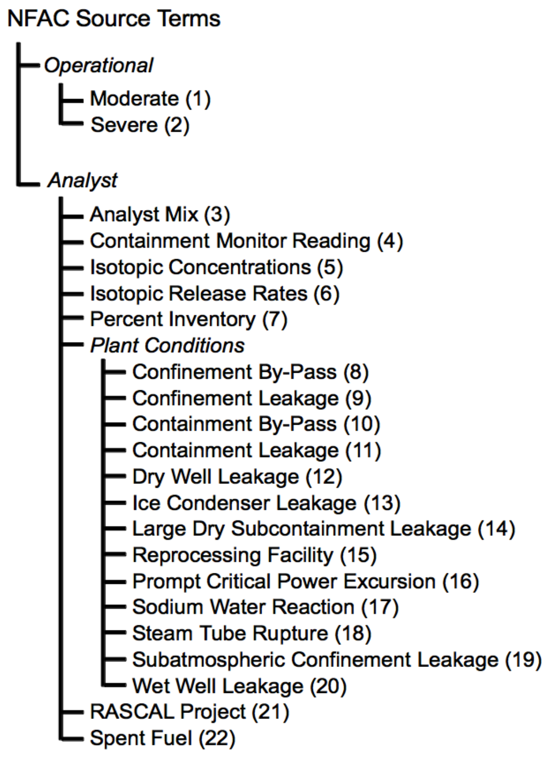

Fig. 1: NFAC Source Terms.

2. TEST APPROACH

Testing is divided into two paths: automated tests and interactive tests. Six automated JUnit4 test suites comprising 87 individual tests have been built and are described in detail below. Each individual test includes one or more assertions challenging specific results or consequences designed to exercise a key functional capability. The granularity of the capability being tested varies by test from an individual component method to an aggregate or composite test involving multiple components and methods. Success or failure of automated tests is an output of the test framework. They are run as unattended batch processes.

The automated tests aggregate information written by NFAC components to the log file. The log output contains all the data regarding activity table inputs fed to RTH and thus can be compared with baseline logs for equivalence. This represents a unit test of NFAC. However, in some test cases we go further to compare the radfile produced by RTH with a baseline radfile, providing in effect an integration test with RTH components. Since the HPAC architecture is such that NFAC is dependent on RTH for integration within HPAC, we feel this level of integration testing is justified in conjunction with corresponding NFAC unit tests that verify NFAC-produced inputs in the NFAC-RTH interface.

The GUI is not conducive to automated tests and thus must be tested interactively. All capabilities in the GUI are tested explicitly as described below.

3. NFACTEST OSGI BUNDLE

NFAC automated tests have been inherited from prior HPAC versions where testing could rely on Ant JUnit tasks. However, HPACs OSGi environment has made it necessary for NFAC to provide its own custom testing framework that allows NFAC's JUnit4 tests to run under OSGi. Further, NFAC's custom framework, the nfactest bundle, includes components for reading log files and radfiles so they can be compared as test assertions. These test framework components are described briefly below and listed as source in appendices.

3.1 NFACLOG

In HPAC-5.3, logging was added to NFAC components to document the activity table inputs fed to RTH for the ultimate generation of the radfile used by SCIPUFF. This was enhanced in HPAC-6.1 to be sufficient for extraction and comparison as part of a test assertion. Note this logging capability was built using java.util.logging and has been extended with the same logging framework. So much effort has been invested in this capability that it is now an integral, unremovable, part of NFAC. Fortunately, with version 6.2 HPAC has reverted to use of java.util.logging as the preferred logging mechanism. The NfacLog class provides the ability to read a log file, extract the activity table inputs logged by NFAC, encapsulate them, and provide equality comparison between NfacLog instances. Refer to Appendix B. for a source listing of this component.

3.2 RADFILE

This component was added to read radfiles generated by RTH and provide equality comparison between instances. The RadFile class accounts for material names that are specific to an individual radfile, matching material names between files so that comparison is possible. Refer to Appendix C. for a source list of this components.

3.3 NFACJUNIT4ADAPTER

This class provides the mechanics of finding and running JUnit4 test suites and cases, while using the OSGi environment for class loading. It borrows heavily from the Ant JUnit tasks and provides similar reporting output. The NfacJUnit4Adapter class uses the Junit4 API to run tests and receive events related to suite and individual test case runs. Refer to Appendix A. for a source listing.

4. AUTOMATED TESTS

4.1 EXTERNALFILEFINDERTEST

This suite is composed of four individual tests to verify correct function of NFAC's PropertyFinder class used to identify and locate ExternalFile instances to be (re)stored for a project. PropertyFinder instances are used by NfacImpl instances in exportFiles() and importFiles() methods now required for correct HPAC project export and import operations, respectively. Tests in alphabetical order are summarized below.

NFAC components tested:

mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDef
mil.dtra.hpac.models.nfac.CAcomp.data.NfacIncident
mil.dtra.hpac.models.nfac.CAcomp.data.PropertyFinder

mil.dtra.hpac.models.nfac.CAcomp.impl.NfacImpl

Test: testAllIncidents

Loads a project containing multiple NfacImpl instances, assigns a custom inventory with an ExternalFile reference to each instance, and invokes the finder() method of a PropertyFinder instance to locate the ExternalFiles. Verifies all ExternalFiles are found.

Test: testFileReleases

Loads a project containing FileRelease instances and ensures all associated ExternalFiles are found.

Test: testNfacImpl

Tests correct behavior of the NfacImpl.exportFiles() method.

Test: testProject

Loads a project with an NFAC incident with a custom inventory and verifies that the findExternalFiles() method of PropertyFinder behaves correctly.

4.2 MODELDEFSTEST

This suite contains 42 individual tests designed to verify proper function of each of the NFAC source models. For each test there is a corresponding HPAC 6 project (*.hpac6*) file or a project export (*.zip*) file. For most tests a radfile (*.rad*) from the baseline project execution is a fixture, and when appropriate a log file (*.log*) is an additional fixture. A ProjectManager instance is used to open the project file, and a DispersionManager instance is used to calculate the project via calls to startDispersion() and waitForProcess(). The project is then saved via ProjectManager.saveProject(), and the radfile is extracted from the project archive (*.hzip*) file or directory, whichever exists. The radfile resulting from the test run and the baseline radfile are read into RadFile instances and compared with RadFile.equals(). For those tests including a log file fixture, the fixture and log file resulting from the run are read into NfacLog objects and compared with the NfacLog.equals() method. Exact matches determine a successful test. Any difference in RadFile or NfacLog objects represents a test failure. Tests in alphabetical order are summarized below.

NFAC components tested:

mil.dtra.hpac.models.nfac.CAcomp.data.ActivityHash
mil.dtra.hpac.models.nfac.CAcomp.data.AmbClient
mil.dtra.hpac.models.nfac.CAcomp.data.Element
mil.dtra.hpac.models.nfac.CAcomp.data.Facility
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDB
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDBMgr
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDef
mil.dtra.hpac.models.nfac.CAcomp.data.MaterialMode
mil.dtra.hpac.models.nfac.CAcomp.data.ModelDefUtils
mil.dtra.hpac.models.nfac.CAcomp.data.ModelTimes
mil.dtra.hpac.models.nfac.CAcomp.data.ModerateModel
mil.dtra.hpac.models.nfac.CAcomp.data.NfacIncident
mil.dtra.hpac.models.nfac.CAcomp.data.NfacIncidentMgr
mil.dtra.hpac.models.nfac.CAcomp.data.NfacRelease
mil.dtra.hpac.models.nfac.CAcomp.data.Options
mil.dtra.hpac.models.nfac.CAcomp.data.ParticleGroups
mil.dtra.hpac.models.nfac.CAcomp.data.ParticleGroupsMgr
mil.dtra.hpac.models.nfac.CAcomp.data.ReproInventory
mil.dtra.hpac.models.nfac.CAcomp.data.SevereModel

mil.dtra.hpac.models.nfac.CAcomp.data.analyst.AnalystMix
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.AnalystModelUtils
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.ConcentrationUnits
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.ContainmentMonitorReading
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.IsotopeReleases
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.IsotopeValue
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.IsotopicConcentrations
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.IsotopicReleaseRates
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PIRSubReleaseMode
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PercentInventory
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PercentInventoryConstants
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PercentInventoryRelease
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PercentInventoryReleaseTree
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.RascalFileReader
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.RascalProject
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.RascalXmlReader
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.ReleaseRate
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.ReleaseRateUnits
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.SpentFuel

mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCConfinementByPass

mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCConfinementLeakage
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCContainmentByPass
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCContainmentLeakage
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCDryWellLeakage
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCIceCondenserContainment
...models.nfac.CAcomp.data.analyst.plantcond.PCLargeDrySubContainmentLeakage
...models.nfac.CAcomp.data.analyst.plantcond.PCPromptCriticalPowerExcursion
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCReproFacility
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCWaterReaction
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCSteamTubeRupture
...models.nfac.CAcomp.data.analyst.plantcond.PCSubatmosphericConfinementLeakage
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.plantcond.PCWetWellLeakaage

mil.dtra.hpac.models.nfac.CAcomp.impl.MelcorFraction
mil.dtra.hpac.models.nfac.CAcomp.impl.MelcorFractions
mil.dtra.hpac.models.nfac.CAcomp.impl.MelcorReleaseFractionsFile
mil.dtra.hpac.models.nfac.CAcomp.impl.NfacImpl
mil.dtra.hpac.models.nfac.CAcomp.impl.PercentInventoryFileMgr
mil.dtra.hpac.models.nfac.CAcomp.impl.RepoData
mil.dtra.hpac.models.nfac.CAcomp.impl.ReproMod
mil.dtra.hpac.models.nfac.CAcomp.impl.StcalcClient

Test: test10682GroupsExplicit

Tests the new capabilities added for SPCR 10682. Two material processing modes are now supported: the original *Simple* scheme where depositor and non-depositor materials are created for each release, and a new *Groups* mode where separate gas materials are created representing particle sizes and associated gas deposition velocities. Isotopes are apportioned according to ratios established for each MELCOR group. Two sets of apportionment ratios are provided, one for BWR reactors and another for PWRs (also applied to VVER). This test ensures the apportionment is correct and the corresponding materials are created with the Groups material mode and the Explicit percent inventory sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Groups
Sub-release Mode:	Explicit
Source Term:	Percent Inventory
Shutdown Duration:	0
Release Duration:	7.0 h

Releases:

#	Duration	Percentages by Group
0	0.5 h	NobleGas=0.01, AlkaMetal=0.6, Chalcogen=0.35
1	2.0 h	NobleGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45
2	4.5 h	NobelGas=0.8, AlkaMetal=0.9, Chalcogen=0.45

Test: test10682GroupsNone

Tests proper activity apportionment and material creation for the Groups material mode and None percent inventory sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Groups
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	0
Release Duration:	7.0 h

Releases:

#	Duration	Percentages by Group
0	0.5 h	NobleGas=0.01, AlkaMetal=0.6, Chalcogen=0.35
1	2.0 h	NobleGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45
2	4.5 h	NobelGas=0.8, AlkaMetal=0.9, Chalcogen=0.45

Test: test10682PwrGroupsNone

Tests proper activity apportionment and material creation for the Groups material mode and None percent inventory sub-release mode.

Facility:	Arkansas One-1 (PWR)
Operating Power:	2550.0 MWt
Material Mode:	Groups
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	1.0 h
Release Duration:	7.0 h

Releases:

#	Duration	Percentages by Group
0	1.0 h	0
1	5.0 h	NobleGas=3.0, AlkaMetal=0.003, Halogens=0.003, Chalcogen=0.03, Platinoid=1.68e-6, Tetravalent=4.0e-7, Trivalent=6.0e-7
2	2.0 h	NobleGas=17, AlkaMetal=0.07, AlkaEarth=8.95e-5, Halogens=0.089, Chalcogen=0.64,Platinoid=6.2e-6, Tetravalent=7.0e-7, Trivalent=1.6e-6

Test: test10682RascalGroupsNone

Tests proper activity apportionment and material creation for a RASCAL source term and the Groups material mode.

Facility:	Robinson-2 (PWR)
Operating Power:	2295.0 MWt
Material Mode:	Groups
Source Term:	Rascal Project
Units:	Ci
Release Times:	0.0, 2.0, 4.0, 6.0, 8.0 h
Isotopes:	Ba-140, Ce-141, Ce-144, Cs-137, I-131, Kr-85, La-140, Mo-99, Nb-95, Nd-147, Pr-143, Pu-239, Ru-103, Ru-106, Sr-89, Sr-90, Te-129, Xe-133, Xe-135, Y-91, Zr-95

Test: test10682RascalSimpleNone

Tests proper activity apportionment and material creation for a RASCAL source term and the Groups material mode.

Facility:	Robinson-2 (PWR)
Operating Power:	2295.0 MWt
Material Mode:	Simple
Source Term:	Rascal Project
Units:	Ci
Release Times:	0.0, 2.0, 4.0, 6.0, 8.0 h
Isotopes:	Ba-140, Ce-141, Ce-144, Cs-137, I-131, Kr-85, La-140, Mo-99, Nb-95, Nd-147, Pr-143, Pu-239, Ru-103, Ru-106, Sr-89, Sr-90, Te-129, Xe-133, Xe-135, Y-91, Zr-95

Test: test10682SimpleExplicit

Tests that activity apportionment and per-particle-size material creation do not occur for a scenario using the Explicit percent inventory sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	Explicit
Source Term:	Percent Inventory
Shutdown Duration:	0
Release Duration:	7.0 h
Releases:	

#	Duration	Percentages by Group
0	0.5 h	NobleGas=0.01, AlkaMetal=0.6, Chalcogen=0.35,
1	2.0 h	NobleGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45,
2	4.5 h	NobleGas=0.8, AlkaMetal=0.9, Chalcogen=0.45

Test: test10682SimpleNone

Tests that activity apportionment and per-particle-size material creation do not occur for a scenario using the None percent inventory sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	0
Release Duration:	7.0 h
Releases:	

#	Duration	Percentages by Group
0	0.5 h	NobleGas=0.01, AlkaMetal=0.6, Chalcogen=0.35,
1	2.0 h	NobleGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45,
2	4.5 h	NobleGas=0.8, AlkaMetal=0.9, Chalcogen=0.45

Test: test10682UnsupportedGroupsNone

Tests that activity apportionment and per-particle-size material creation do not occur even though requested via the Groups material mode for an unsupported facility type and the None percent inventory sub-release mode.

Facility:	Hartlepool-1 (AGR)
Operating Power:	1785.0 MWt
Material Mode:	*disabled*
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	1.0 h
Release Duration:	7.0 h
Releases:	

#	Duration	Percentages by Group
0	1.0 h	0
1	2.0 h	NobleGas=3.0, AlkaMetal=0.003, Halogens=0.003, Chalcogen=0.03, Platinoid=1.68e-6, Tetravalent=4.0e-7, Trivalent=6.0e-7
2	5.0 h	NobleGas=17, AlkaMetal=0.07, AlkaEarth=8.95e-5, Chalcogen=0.64, Platinoid=6.2e-6, Tetravalent=7.0e-7, Trivalent=1.6e-6

Test: test10682VverGroupsNone

Tests proper activity apportionment and material creation occurs for a VVER facility based on the model for a PWR using the Groups material mode and the None percent inventory sub-release mode.

Facility:	Zaporozhe-1 (VVER-1000)
Operating Power:	2850.0 MWt
Material Mode:	Groups
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	1.0 h
Release Duration:	7.0 h
Releases:	

#	Duration	Percentages by Group
0	1.0 h	0
1	2.0 h	NobleGas=3.0, AlkaMetal=0.003, Halogens=0.003, Chalcogen=0.03, Platinoid=1.68e-6, Tetravalent=4.0e-7, Trivalent=6.0e-7
2	5.0 h	NobleGas=17, AlkaMetal=0.07, AlkaEarth=8.95e-5, Chalcogen=0.64, Platinoid=6.2e-6, Tetravalent=7.0e-7, Trivalent=1.6e-6

Test: test11653

Explicitly tests the fix for SPCR 11653 to ensure a custom inventory is accounted for when determining the available plant conditions source terms.

Facility:	Arkansas One-1 (PWR)
Operating Power:	2550.0 MWt
Custom Inventory:	Created from kewaunee.f71 ORIGEN file
Material Mode:	Simple
Source Term:	Mix Specified by Analyst
Gross Release Rate:	1.0 Ci/s
Release Percentages:	

Nobel Gases:	98.0%
Halogens:	2.0%

Test: testAnalystMix

Tests an AnalystMix model instance with the following properties:

Facility:	Brunswick-1 (BWR Mk-1)		
Operating Power:	2949.0 MWt		
Material Mode:	Simple	Nobel Gases:	98.0%
Source Term:	Mix Specified by Analyst	Halogens:	2.0%
Gross Release Rate:	100.0 Ci/s		
Release Percentages:			

Test: testContainmentMonitorReading

Tests a ContainmentMonitorReading model instance with the following properties:

Facility:	Brunswick-1 (BWR Mk-1)
Operating Power:	2949.0 MWt
Material Mode:	Simple
Source Term:	Containment Monitor Reading
Leak Rate:	10%/h
Monitor Location:	Wet Well
Monitor Reading:	25.0 R/h
Release Path:	Unfiltered
Sprays:	Off

Test: testIsotopicConcentrations

Tests a IsotopicConcentrations model instance with the following properties:

Facility:	Brunswick-1 (BWR Mk-1)		
Operating Power:	2949.0 MWt	Cs-130:	130.0
Material Mode:	Simple	Cs-131:	131.0
Source Term:	Isotopic Concentrations	I-130:	130.0
Concentration Units:	kCi/cc	I-132:	132.0
Release Rate:	20.0 cc/s	I-133:	133.0
Isotope Values:			

Test: testIsotopicReleaseRates

Tests a IsotopicReleaseRates model instance with the following properties:

		Cs-130:	130.0
Facility:	Brunswick-1 (BWR Mk-1)	Cs-131:	131.0
Operating Power:	2949.0 MWt	Cs-132:	132.0
Material Mode:	Simple	Cs-134:	134.0
Source Term:	Isotopic Release Rates	I-130:	130.0
Release Units:	kCi/s	I-131:	131.0
Isotope Values:		I-132:	132.0
		I-133:	133.0

Test: testModerate

Tests an operational ModerateModel instance with properties:

Facility:	Brunswick-1 (BWR Mk-1)
Operating Power:	2949.0 MWt

Test: testMultiFive

Tests a project with five NFAC incidents.

Facility:	Calvert Cliffs-1 (PWR)
Operating Power:	2535.0 MWt
Materal Mode:	Simple
Source Term:	Steam Generator Tube Rupture (Coolant)
Partitioned Generator:	false
Release Rate:	1 Tube (35%/h)
Release Source:	Steam Jet Air Ejector

Facility:	Limerick-1 (BWR Mk-2)
Operating Power:	3165.0 MWt
Materal Mode:	Simple
Source Term:	Dry Well Leakage/Failure (BWR Containment)
Core Condition:	Vessel Melt Through
Leak Rate:	50%/h (release duration=2.0 h)
Filtered Release Path:	false
Sprays On:	false

Facility:	Oyster Creek (BWR Mk-1)
Operating Power:	1950.0 MWt
Materal Mode:	Simple
Source Term:	Containment Monitor Reading
Leak Rate:	50%/h (release duration=2.0 h)
Monitor Location:	Wet Well
Monitor Reading:	20.0 R/h
Release Path:	Unfiltered
Sprays:	Off

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Materal Mode:	Simple
Source Term:	Spent Fuel/Spent Fuel Pool
Fuel Condition:	Fuel Cladding Failure
Number of Batches:	1
Release Path:	Unfiltered
Sprays:	Off
Time Last Batch in Pool:	*same as ReleaseToContainment*

Facility:	Salem-1 (PWR)
Operating Power:	3507.0 MWt
Materal Mode:	Simple
Source Term:	Large, Dry, or Subatmospheric Containment Leakage/Failure
Core Condition:	Gap Release
Filtered Release Path:	false
Leak Rate:	50%/h (release duration=2.0 h)
Sprays On:	false

Test: testPCConfinementByPass

Tests a PCConfinementByPass plant conditions model instance with properties:

Facility:	Hartlepool-1 (AGR)
Operating Power:	1785.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Bypass of Confinement
Core Condition:	Gap Release
Leak Rate:	25%/h (release duration=4.0 h)

Test: testPCConfinementLeakage

Tests a PCConfinementLeakage plant conditions model instance with properties:

Facility:	Novovoronezh-3 (VVER-400/230)
Operating Power:	1155.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Confinement Leakage/Failure
Core Condition:	Gap Release
Leak Rate:	50%/h (release duration=2.0 h)

Test: testPCContainmentByPass

Tests a PCContainmentByPass plant conditions model instance with properties:

Facility:	Hamaoka-3 (BWR Mk-1)
Operating Power:	3168.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Bypass of Containment
Core Condition:	Gap Release
Leak Rate:	100%/h (release duration=1.0 h)
Filtered Release Path:	false

Test: testPCContainmentLeakage

Tests a PCContainmentLeakage plant conditions model instance with properties:

Facility:	Beloyarski-3 (BN-600) (LMFBR)
Operating Power:	1680.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Containment Leakage/Failure
Core Condition:	Gap Release
Leak Rate:	50%/h (release duration=2.0 h)
Filtered Release Path:	false

Test: testPCDryWellLeakage

Tests a PCDryWellLeakage plant conditions model instance with properties:

Facility:	Hamaoka-3 (BWR Mk-1)
Operating Power:	3168.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Dry Well Leakage/Failure (BWR Containment)
Core Condition:	Gap Release
Filtered Release Path:	false
Leak Rate:	100%/h (release duration=1.0 h)
Sprays On:	false

Test: testPCIceCondenserContainmentCoreDamage

Tests a PCIceCondenserContainment plant conditions model instance with properties:

Facility:	Watts Bar-1 (PWR)
Operating Power:	3465.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Ice Condenser Containment Leakage/Failure
Core Condition:	In-Vessel Severe Core Damage
Fans On:	true
Filtered Release Path:	true
Ice Bed Exhausted:	true
Leak Rate:	50%/h (release duration=2.0 h)
Sprays On:	true

Test: testPCIceCondenserContainmentGapRelease

Tests a PCIceCondenserContainment plant conditions model instance with properties:

Facility:	Watts Bar-1 (PWR)
Operating Power:	3465.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Ice Condenser Containment Leakage/Failure
Core Condition:	Gap Release
Fans On:	false
Filtered Release Path:	false
Ice Bed Exhausted:	false
Leak Rate:	50%/h (release duration=2.0 h)
Sprays On:	false

Test: testPCIceCondenserContainmentMeltThrough

Tests a PCIceCondenserContainment plant conditions model instance with properties:

Facility:	Watts Bar-1 (PWR)
Operating Power:	3465.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Ice Condenser Containment Leakage/Failure
Core Condition:	Vessel Melt Through
Fans On:	false
Filtered Release Path:	false
Ice Bed Exhausted:	false
Leak Rate:	50%/h (release duration=2.0 h)
Sprays On:	false

Test: testPCLargeDrySubcontainmentLeakage

Tests a PCLargeDrySubcontainmentLeakage plant conditions model instance with properties:

Facility:	Beaver Valley-1 (PWR)
Operating Power:	2733.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Large, Dry, or Subatmospheric Containment Leakage/Failure
Core Condition:	Gap Release
Filtered Release Path:	false
Leak Rate:	100%/h (release duration=1.0 h)
Sprays On:	false

Test: testPCPromptCriticalPowerExcursion

Tests a PCPromptCriticalPowerExcursion plant conditions model instance with properties:

Facility:	Bilibino Unit A (RBMK)
Operating Power:	33.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Prompt Critical Power Excursion
Core Involvement:	Total

Test: testPCReproFacility

Tests a PCReproFacility plant conditions model instance with properties:

Facility:	Savannah River (Reprocessing)
Material Mode:	*disabled*
Source Term:	Plant Conditions: Reprocessing Facility
Plant Throughput:	1.0 tonnes/day
Release Rate:	100%/h (release duration=1.0 h)
Release Fraction by Component:	

Aqueous Waste Treatment:	0.5
Dissolution: :	1.0
Feed Adjustment and Accountability: :	1.0
Pu Recovery: :	0.5
Solvent Treatment: :	0.5
U-Pu Co-Decontamination, Partitioning, and U Purification:	1.0

Test: testPCSodiumWaterReaction

Tests a PCSodiumWaterReaction plant conditions model instance with properties:

Facility:	Phenix (LMFBR)
Custom Inventory:	stlauren_1.avc
Operating Power:	699.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Sodium-Water Reaction
Core Condition:	Gap Release
Leak Rate:	50%/h (release duration=2.0 h)

Test: testPCSteamTubeRuptureCoreDamage

Tests a PCSteamTubeRupture plant conditions model instance with properties:

Facility:	Dungeness B-1 (AGR)
Operating Power:	1560.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Steam Generator Tube Rupture (Coolant)
Coolant Concentration:	In-Vessel Severe Core Damage
Partitioned Generator:	false
Release Rate:	1 Tube (35%/h)
Release Source:	Safety Valve

Test: testPCSteamTubeRuptureGapRelease

Tests a PCSteamTubeRupture plant conditions model instance with properties:

Facility:	Dungeness B-1 (AGR)
Operating Power:	1560.0 MWt
Material Mode:	*disabled*
Source Term:	Plant Conditions: Steam Generator Tube Rupture (Coolant)
Coolant Concentration:	Gap Release
Partitioned Generator:	false
Release Rate:	1 Tube (35%/h)
Release Source:	Safety Valve

Test: testPCSubatmosphericConfinementLeakage

Tests a PCSubatmosphericConfinementLeakage plant conditions model instance with properties:

Facility:	Rovno-1 (VVER-440/213)
Operating Power:	1143.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Subatmospheric Confinement Leakage/Failure
Core Condition:	Gap Release
Leak Rate:	100%/h (release duration=1.0 h)
Pool Suppression System:	true

Test: testPCWetWellLeakage

Tests a PCWetWellLeakage plant conditions model instance with properties:

Facility:	Hamaoka-3 (BWR Mk-1)
Operating Power:	3168.0 MWt
Material Mode:	Simple
Source Term:	Plant Conditions: Wet Well Leakage/Failure (BWR Containment)
Core Condition:	Gap Release
Leak Rate:	100%/h (release duration=1.0 h)
Filtered Release Path:	false
Wet Well:	Saturated

Test: testPercentInventoryNoShutdownAsIs

Tests a PercentInventory model instance with no shutdown duration defined in the releases and no override of the model event times.

Facility:	Watts Bar-1 (PWR)
Operating Power:	3465.0 MWt
Material Mode:	Simple
Sub-release Mode:	Average
Source Term:	Percent Inventory
Shutdown Duration:	0
Release Duration:	44.9 h
Releases:	

#	Duration	Percentages by Group
0	1.9 h	NobleGas=3.0, AlkaMetal=0.003, AlkaEarth=1e-5, Halogens=0.003, Chalcogen=0.03, Platinoid=1.2e-7
1	32.0 h	NobleGas=3.0, AlkaMetal=0.003, Halogens=0.003, Chalcogen=0.03, Platinoid=1.68e-6, Tetravalent=4.0e-7, Trivalent=6.0e-7
2	11.0 h	NobleGas=17, AlkaMetal=0.07, AlkaEarth=8.95e-5, Halogens=8.9e-2, Chalcogen=0.64, Platinoid=6.2e-6, Tetravalent=7.0e-7, Trivalent=1.6e-6

Test: testPercentInventoryNoShutdownForced

Tests a PercentInventory model instance with no shutdown duration defined in the releases, but model event times are overridden to force a shutdown time.

Facility:	Watts Bar-1 (PWR)
Operating Power:	3465.0 MWt
Material Mode:	Simple
Sub-release Mode:	Average
Source Term:	Percent Inventory
Shutdown Duration:	10 m
Release Duration:	44.9 h
Releases:	

#	Duration	Percentages by Group
0	1.9 h	NobleGas=3.0, AlkaMetal=0.003, AlkaEarth=1e-5, Halogens=0.003, Chalcogen=0.03, Platinoid=1.2e-7
1	32.0 h	NobleGas=3.0, AlkaMetal=0.003, Halogens=0.003, Chalcogen=0.03, Platinoid=1.68e-6, Tetravalent=4.0e-7, Trivalent=6.0e-7
2	11.0 h	NobleGas=17, AlkaMetal=0.07, AlkaEarth=8.95e-5, Halogens=8.9e-2, Chalcogen=0.64, Platinoid=6.2e-6, Tetravalent=7.0e-7, Trivalent=1.6e-6

Test: testPercentInventoryWithShutdownAsIs

Tests a PercentInventory model instance with a shutdown duration defined in the releases and no override of the model event times.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	Average
Source Term:	Percent Inventory
Shutdown Duration:	41.7 m
Release Duration:	11.3 h
Releases:	

#	Duration	Percentages by Group
0	41.7 m	0
1	41.7 m	NobelGas=0.01, AlkaMetal=0.6, Chalcogen=0.35
2	83.33 m	NobelGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45
3	250.0 m	NobelGas=0.8, AlkaMetal=0.9, Chalcogen=0.45
4	303.0 m	NobelGas=0.3, AlkaMetal=0.1, Chalcogen=0.22, LessVolatile=0.27

Test: testPercentInventoryWithShutdownForced

Tests a PercentInventory model instance with a shutdown duration defined in the releases, but model event times are overridden to force no shutdown time.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	Average
Source Term:	Percent Inventory
Shutdown Duration:	41.7 m
Release Duration:	11.3 h
Releases:	

#	Duration	Percentages by Group
0	41.7 m	0
1	41.7 m	NobelGas=0.01, AlkaMetal=0.6, Chalcogen=0.35
2	83.33 m	NobelGas=0.09, AlkaMetal=1.7, AlkaEarth=0.03, Chalcogen=1.45
3	250.0 m	NobelGas=0.8, AlkaMetal=0.9, Chalcogen=0.45
4	303.0 m	NobelGas=0.3, AlkaMetal=0.1, Chalcogen=0.22, LessVolatile=0.27

Test: testPIRSubReleasesAverage

Tests a PercentInventory model instance with the Average PIR sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	Average
Source Term:	Percent Inventory
Shutdown Duration:	30.0 m
Release Duration:	48.0 h
Releases:	

#	Duration	Percentages by Group
0	48.0 h	NobelGas=100

Test: testPIRSubReleasesExplicit

Tests a PercentInventory model instance with the Explicit PIR sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	Explicit
Source Term:	Percent Inventory
Shutdown Duration:	30.0 m
Release Duration:	48.0 h
Releases:	

#	Duration	Percentages by Group
0	48.0 h	NobelGas=100

Test: testPIRSubReleasesNone

Tests a PercentInventory model instance with the None PIR sub-release mode.

Facility:	Peach Bottom-2 (BWR Mk-1)
Operating Power:	3414.0 MWt
Material Mode:	Simple
Sub-release Mode:	None
Source Term:	Percent Inventory
Shutdown Duration:	30.0 m
Release Duration:	48.0 h
Releases:	

#	Duration	Percentages by Group
0	48.0 h	NobelGas=100

Test: testRascalProject

Tests a RascalProject model instance with properties:

Facility:	Brunswick-1 (BWR Mk-1)
Operating Power:	2949.0 MWt
Material Mode:	Simple
Source Term:	Rascal Project
Units:	Ci
Release Times:	0.0, 0.25, 0.5, 0.75, ..., 8.0 h
Isotopes:	Ba-139, Ba-140, Cs-134, Cs-136, Cs-137, Cs-138, I-131, I-132, I-133, I-134, I-135, Kr-83m, Kr-85, Kr-85m, Kr-87, Kr-88, La-140, Mo-99, Rb-86, Rb-88, Rh-103, Rh-105, Ru-103, Ru-105, Ru-106, Sb-127, Sb-129, Sr-89, Sr-90, Sr-91, Sr-92, Tc-99m, Tc-127, Tc-127m, Tc-129, Tc-129m, Tc-131, Tc-131m, Tc-132, Xc-131m, Xc-133, Xc-133m, Xc-135, Xc-135m, Xc-138, Y-90, Y-91, Y-91m, Y-92

Test: testSevere

Tests an operational SevereModel instance with properties:

Facility:	Brunswick-1 (BWR Mk-1)
Operating Power:	2949.0 MWt

Test: testSpentFuel

Tests an operational SpentFuel instance with properties:

Facility:	Brunswick-1 (BWR Mk-1)
Operating Power:	2949.0 MWt
Source Term:	Spent Fuel/Spent Fuel Pool
Fuel Condition:	Fuel Cladding Failure
Number of Batches:	1
Release Path:	Unfiltered
Sprays:	Off
Time Last Batch in Pool:	*6 months prior to ReleaseToContainment*

4.3 MODELTIMESTEST

One of the most critical aspects of NFAC processing is correct management of event times within a scenario. There are six possible events whose times define or are defined by associated durations:

Event	Associated Duration	From Event
Shutdown / Start of Decay		
Release to Containment	Containment	Start of Decay
Release to Environment	Shutdown	Start of Decay
	Holdup	Release to Containment
End of Release	Release	Release to Environment
End of Dispersion	Dispersion	End of Release
End of Exposure	Exposure	End of Dispersion

For some NFAC source models, individual events have no meaning and thus are ignored and not displayed in the GUI. Some times are computed directly from the model definition and thus are made uneditable (although displayed) in the GUI. Which times are displayed and/or editable are determined based on the model, and uneditable times must be computed based on the model parameters. Finally, uneditable times can be forced to be editable via the Advanced Mode toggle button in the When tab. The purpose of this test suite is to exercise event times management against various models and conditions. The eight tests are described below. Each unique times management class is tested.

NFAC components tested:

mil.dtra.hpac.models.nfac.CAcomp.data.ModelTimes

mil.dtra.hpac.models.nfac.CAcomp.data.times.AnalystMixTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.IsotopicConcentrationsTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.IsotopicReleaseRatesTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.ModelTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.ModelTimesMgrFactory
mil.dtra.hpac.models.nfac.CAcomp.data.times.OperationalTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.PCContainmentByPassTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.PercentInventoryTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.RascalProjectTimesMgr
mil.dtra.hpac.models.nfac.CAcomp.data.times.SpentFuelTimesMgr

Test: testAnalystMix

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns an AnalystMixTimesMgr instance for an AnalystMix model.
- The default times based on test input times are correctly set by the AnalystMixTimesMgr instance.
- After explicit settings of editable times, the Release to Environment and Release to Containment times are equivalent.

Test: testIsotopicConcentrations

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns an IsotopicConcentrationsTimesMgr instance for an IsotopicConcentrations model.
- The default times based on test input times are correctly set by the IsotopicConcentrationsTimesMgr instance.
- After explicit settings of editable times, the Start of Decay, Release to Containment, and Release to Environment times are equivalent.

Test: testIsotopicReleaseRates

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns an IsotopicReleaseRatesTimesMgr instance for an IsotopicReleaseRates model.
- The default times based on test input times are correctly set by the IsotopicReleaseRatesTimesMgr instance.
- After explicit settings of editable times, the Start of Decay, Release to Containment, and Release to Environment times are equivalent.

Test: testPCContainmentByPass

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns a PCContainmentByPassTimesMgr instance for an PCContainmentByPass model.
- The default times based on test input times are correctly set by the PCContainmentByPassTimesMgr instance.
- After explicit settings of editable times, the Start of Decay and Release to Containment times are equivalent, and the Shutdown, Release, Dispersion, and Exposure durations are all correct.

Test: testPercentInventoryNoShutdown

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns a PercentInventoryTimesMgr instance for a PercentInventory model.
- After explicit setting of editable times, the Release to Containment and Release to Environment times are equivalent, the Shutdown Duration is zero, and the Release, Dispersion, and Exposure durations are correct.
- After explicit setting of editable times in lenient mode, the Release to Containment and Release to Environment times are equivalent, and the Shutdown, Release, Dispersion, and Exposure durations are correct for the input times.

Test: testPercentInventoryShutdown

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns a PercentInventoryTimesMgr instance for a PercentInventory model.
- After explicit setting of editable times, the Release to Containment and Release to Environment times are equivalent, and the Shutdown, Release, Dispersion, and Exposure durations are correct.
- After explicit setting of editable times in lenient mode, the Release to Containment and Release to Environment times are equivalent, and the Shutdown, Release, Dispersion, and Exposure durations are correct for the input times.

Test: testRascalProject

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns a RascalProjectTimesMgr instance for an RascalProject model.
- After explicit setting of editable times, the Start of Decay, Release to Containment and Release to Environment times are equivalent.

Test: testSpentFuel

This test verifies the following:

- The ModelTimesMgrFactory.getInstance() method correctly returns a SpentFuelTimesMgr instance for an SpentFuel model.
- After explicit setting of editable times, the Start of Decay and Release to Containment times are equivalent, and the Holdup, Shutdown, Release, Dispersion, and Exposure durations are correct.

4.4 PARTICLEGROUPSMGRTEST

The major enhancement for NFAC between versions 6.2 and 6.3 was the addition of a new mode for processing materials. Previously, NFAC represented each activity vector with two materials, a depositor and an optional non-depositor for any noble gas isotopes. As per SPCR 10682, in addition to the Simple depositor/non-depositor approach, a Groups mode was added in which up to eleven materials can be created for each release. Each material represents a particle size group with an associated gas deposition velocity. Tables specifying apportionments into the particle size groups for each MELCOR group are used to assign activities in each activity vector by MELCOR group. Apportionment tables for supported facility types are specified in the NFAC *defaults.properties* data file.

This test suite is comprised of seven tests that ensure the apportionment tables are properly read and activity apportionment and material creation are correctly processsed.

NFAC components tested:

mil.dtra.hpac.models.nfac.CAcomp.data.ParticleGroups
mil.dtra.hpac.models.nfac.CAcomp.data.ParticleGroupsMgr

Test: testBWRCounts

Checks that the number of particle size groups, MELCOR groups, and table entries for each MELCOR group associated with the BWR facility type, as read from *defaults.properties*, are correct.

Test: testBWRValues

Checks that the BWR table entries giving apportionments for each MELCOR group and particle group, as read from *defaults.properties* are correct.

Test: testPWRCounts

Checks that the number of particle size groups, MELCOR groups, and table entries for each MELCOR group associated with the PWR facility type, as read from *defaults.properties*, are correct.

Test: testPWRValues

Checks that the PWR table entries giving apportionments for each MELCOR group and particle group, as read from *defaults.properties*, are correct.

Test: testReadCounts

Checks that the number of particle group tables and the list of associated facility types, as read from *defaults.properties*, are correct.

Test: testTypeMap

Verifies that the table associated with each reactor type and subtype are correct if the type/subtype should be supported and are not specified otherwise.

Test: testWriteAndRead

Verifies that the java.util.Properties entries as read from *defaults.properties* can be written and read back with the same resulting deserialized objects.

4.5 SOURCETERMTABLETEST

Another critical piece of NFAC functionality is determination of which of the 22 source models are valid or can be applied to a specific facility definition. A facility definition includes the facility selected from NFAC's facility "database" as well as a reference to any custom inventory file. NFAC's SourceTermTable and PlantConditionTables classes encapsulate the capability of determining available source terms. They are "data driven" in that they read the *source_terms.data* and *plntcond.dat* files, respectively, to drive the determination process.

This test suite includes 25 individual tests to verify that the source model availability process functions correctly.

NFAC components tested:

mil.dtra.hpac.models.nfac.CAcomp.data.Facility
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDB
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDBMgr
mil.dtra.hpac.models.nfac.CAcomp.data.FacilityDef

mil.dtra.hpac.models.nfac.CAcomp.data.analyst.PlantConditionTables
mil.dtra.hpac.models.nfac.CAcomp.data.analyst.SourceTermTable

Test: testAGRCompatibility

Explicitly tests for individual model compatibility via the SourceTermTable.checkCompatability() method. The Dungeness B-1 and Windscale AGR facilities (without custom inventories) are tested against the following models with expected compatibility:

Dungeness B-1

AnalystMix	yes
ContainmentMonitorReading	no
IsotopicConcentrations	yes
IsotopicReleaseRates	yes
ModerateModel	no
PCLargeDrySubContainment	no
PercentInventory	yes
SevereModel	no
SpentFuel	no

Windscale AGR

PCConfinementByPass	yes
PCConfinementLeakage	yes
PCContainmentByPass	no
PCContainmentLeakage	no
PCDryWellLeakage	no
PCIceCondenserContainment	no
PCLargeDrySubContainmentLeakage	no
PCPromptCriticalPowerExcursion	no
PCReproFacility	no
PCSodiumWaterReaction	no
PCSteamTubeRupture	yes
PCSubatmosphericConfinementLeakage	no
PCWetWellLeakage	no

Test: testAGRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following AGR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Dungeness B-1	Windscale AGR[1]	Hunterston B-1[2]

1 Inventory filename set explicitly
2 Inventory filename cleared explicitly

Test: testBWR0SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following BWR type-0 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Leibstadt	Brunsbuettel (kkb)	Leibstadt[2]

2 Inventory filename cleared explicitly

Test: testBWR1SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following BWR type-1 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Santa Maria De Garona	Fukushima-Daiichi-1	Santa Maria De Garona[2]

2 Inventory filename cleared explicitly

Test: testBWR2SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following BWR type-2 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Fukushima-Daini-1	Fukushima-Daini-1[3]	Fukushima-Daini-1[2]

2 Inventory filename cleared explicitly

3 Explicitly set inactive

Test: testBWR3SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following BWR type-3 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Cofrentes	Cofrentes[3]	Cofrentes[2]

2 Inventory filename cleared explicitly

3 Explicitly set inactive

Test: testFBRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following FBR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Monju[1]	KNK II[1]	Monju

1 Inventory filename set explicitly

Test: testGCHWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following GCHWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Monts D'Arree[1]	Monts D'Arree	Monts D'Arree[4]

1 Inventory filename set explicitly
4 Explicitly set active

Test: testGCRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following GCR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Oldbury-1	Oldbury-2	Latina

Test: testHTGRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following HTGR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Peach Bottom[1,4]	Peach Bottom[1]	Peach Bottom

1 Inventory filename set explicitly
4 Explicitly set active

Test: testHWGCRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following HWGCR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
A-1 Bohunice[1,4]	A-1 Bohunice[1]	A-1 Bohunice

1 Inventory filename set explicitly
4 Explicitly set active

Test: testHWLWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following HWLWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Fugen ATR[1,4]	Fugen ATR[1]	Fugen ATR

1 Inventory filename set explicitly
4 Explicitly set active

Test: testLGRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following LGR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Hanford - N[1,4]	Toitsk E	Troitsk F[4]

1 Inventory filename set explicitly
4 Explicitly set active

Test: testLMFBRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following LMFBR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Beloyarsky-3(BN-600)[1]	Beloyarsky-4(BN-800)[1,3]	Beloyarsky-3(BN-600)

1 Inventory filename set explicitly
3 Explicitly set inactive

Test: testLWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following LWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Generic LWR Site	Generic LWR Site[3]	Generic LWR Site[2]

2 Inventory filename cleared explicitly
3 Explicitly set inactive

Test: testPHWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following PHWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Bruce-1	Gentilly-1	Agesta

Test: testPWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following PWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Doel-1	BR-3	Doel-2[2]

2 Inventory filename cleared explicitly

Test: testRBMKSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following RBMK facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Bilibino Unit A	Beloyarsky-1	Bilibino Unit A[2]

2 Inventory filename cleared explicitly

Test: testReproSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following Repro facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Idaho Falls-1	Mol	Idaho Falls-1[2]

2 Inventory filename cleared explicitly

Test: testResearchSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following Research facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
AFRRI TRIGA Reactor	BR-3 (Mol)	ATR

Test: testSGHWRSourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following SGHWR facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Winfrith SGHWR[1,4]	Winfrith SGHWR[1]	Winfrith SGHWR[4]

1 Inventory filename set explicitly
4 Explicitly set active

Test: testVVER0SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following VVER type-0 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Bushehr-1	Bushehr-1[3]	Bushehr-1[2]

2 Inventory filename cleared explicitly
3 Explicitly set inactive

Test: testVVER1SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following VVER type-1 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Metzamor-2	Metzamor-1	Kola-1[2]

2 Inventory filename cleared explicitly

Test: testVVER2SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following VVER type-2 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Dukovany-1	Nord (Greifswald)-5	Dukovany-2[2]

2 Inventory filename cleared explicitly

Test: testVVER3SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following VVER type-3 facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	With inventory + Inactive	No inventory
Kozloduy-5	Kozloduy-6[3]	Kalinin-3[2]

2 Inventory filename cleared explicitly
3 Explicitly set inactive

4.6 SOURCETERMTABLETEST2

This test suite is distinguished from SourceTermTableTest for cases in which no plant condition models apply. For now, this is limited to the new HWRESRC research facility type.

Test: testHWRESRC3SourceTerms

Calls SourceTermTable.isOperationalSupported(), SourceTermTable.getFacilityTerms(), and PlantConditionTables.getPlantConditionTerms() with the following HWRESRC facilities and conditions, verifying the correct source term names are returned:

With inventory + Active	No inventory
IR-40	IR-40[2]

2 Inventory filename cleared explicitly

5. INTERACTIVE TESTS

The purpose of the interactive tests is to exercise each piece of functionality in the Nuclear Facility Incident Edit dialog and its constituent components. Verification of results is by visual inspection of the interface and/or examination of project files after being saved and are recorded in the test descriptions below.

5.1 TOP LEVEL INTERACTIONS

Test: Set Incident Name

Set the incident name via the *Name* edit field.
Verification:

- Incident name correctly (de)serialized from/to the project file.

5.2 WHERE TAB

Test: Select Facility

Select a facility via the *Select* button in the *Facility* group. Examine the facility tree to verify correct representation of the facilities defined in NFAC data files.
Verification:

- The *All Facilities* tree lists all the countries (and generic) represented in the *wcountry.dat* file.
- Each country tree node expands to the proper subtrees (Power Reactor, Reprocessing Facility, and/or Research Reactor) based on facilities available in that country.
- Verify special statuses *decommissioned*, *not yet completed*, and *no inventory* are accurately represented.
- Each leaf node lists the available facilities correctly as per the associated *lwr_??.dat* file.
- The selected facility is correctly represented and (de)serialized from/to the project file.

Test: Select Default Facility Location

Choose the *Facility Location* radio button in the *Location* group for the default facility location.
Verification:

- The coordinate for the facility as stored in the facility database is (de)serialized from/to the project file.

Test: Enter Explicit Facility Location

Selected the *Customized Location* radio button and enter an explicit coordinate for the facility location.
Verification:

- The entered coordinate is (de)serialized from/to the project file as the facility location.

Test: Define Facility Location by Dragging the Incident Icon

With the *Customized Location* radio button selected, drag the incident icon on the map to set the location.
Verification:

- The dragged-to coordinate is (de)serialized from/to the project file as the facility location.

Test: Select the Default Inventory File

Choose the *Default for Facility* radio button in the *Inventory* group.
Verification:

- The inventory file referenced in the facility database entry is (de)serialized from/to the project file.

Test: Select a Custom Inventory File

Choose the *Customized* radio button in the *Inventory* group and specify an inventory (i.e., activity vector or *.avc*) file.
Verification:

- The specified file is (de)serialized from/to the project file as the custom inventory.

Test: Use the ORIGEN File Import Utility to Create a Custom Inventory File

Activate the *Origen Import* button to run the ORIGEN Importer.
Verification:

- After selecting a F71 file, the *ORIGEN File Import* utility appears and displays the cases in the file.
- Activating the *Show File Overview* button brings up an overview windowing showing information about each case in the F71 file.
- After selecting a case and activating the *Show Case Overview* button, a window giving a summary of the selected case appears.

- After selecting a case and activating the *Show Case Details* button, a windowing displaying details of the selected case appears.
- After selecting a case, selecting and entering values in the *Inventory File Units* group, and activating the *Create Inventory File* button, and entering a target filename, an inventory file is created.
- The created inventory file can be specified as the custom inventory for the facility.

5.3 WHAT TAB

Test: Enter an Explicit Operating Power

Change the value of the *Operating Power* field.
Verification:

- The specified operating power is (de)serialized from/to the project file.

Test: Select a Moderate Incident

Choose the *Moderate Incident* radio button.
Verification:

- The Moderate model appropriate for the selected facility is (de)serialized from/to the project file and displayed in the report resulting from activating the *View Source Term* button (compared to the associated *acmxxx.dat* file).

Test: Select a Severe Incident

Choose the *Severe Incident* radio button.
Verification:

- The Severe model appropriate for the selected facility is (de)serialized from/to the project file and displayed in the report resulting from activating the *View Source Term* button (compared to the associated *acs*xxx.dat file).

Test: Define an Isotopic Release Rates Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Isotopic Release Rates* from the *Source Terms Available* list. Click the *Next* button. From the *Define Isotopic Release Rates* dialog, choose *Release Units* and enter values under *Isotope Values*.
Verification:

- The defined isotope values and release units are (de)serialized from/to the project file and displayed in the report resulting from activating the *View Source Term* button.

Test: Define an Isotopic Concentrations Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Isotopic Concentrations* from the *Source Terms Available* list. Click the *Next* button. From the *Define Isotopic Release Rates* dialog, choose *Release Units* and enter values under *Isotope Values*.
Verification:

- The defined isotope values and release units are (de)serialized from/to the project file and displayed in the report resulting from activating the *View Source Term* button.

Test: Define an Percent Inventory Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Percent Inventory* from the *Source Terms Available* list. Click the *Next* button. From the *Define Percent Inventory Releases* dialog, activate the *Load* button and choose *From Moderate Model*. Then, activate the *Load* button and choose *From Severe Model*. Enter explicit values in the release duration and group percentage cells. Activate the *New* button to add a release. Change the duration units via the *Duration Units* combo box. Select a cell in any release column and activate *Delete* to remove that release.

Verification:

- The release definitions for the moderate and severe models are correct for the type of facility *acmxxx.dat* and *acsxxx.dat* files, respectively).
- The (first) totals column is accurately updated after cell edits.
- All duration cell values are converted upon selection of new duration units.
- A new, release column is added when *New* is activated with cell values set to the remainder required to reach 100% release of each group.
- *Delete*d releases are moved.
- The defined releases are (de)serialized from/to the project file and displayed in the report resulting from activating the *View Source Term* button.

Test: Define an Analyst Mix Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Mix Specified by Analyst* from the *Source Terms Available* list. Click the *Next* button. From the *Define Mix Specified by Analyst* dialog, enter a *Gross Release Rate* and various values for the percentages by MELCOR group.

Verification:

- The *Gross Release Rate* value is (de)serialized from/to the project file.
- Entering release percentages totaling more than 100% results in a warning dialog.
- Release percentages for each group are (de)serialized from/to the project file.

Test: Define a Containment Monitor Reading Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Mix Specified by Analyst* from the *Source Terms Available* list. Click the *Next* button. From the *Define Containment Monitor Reading Incident* dialog, enter various values for: *Representative Operating Power*, *Monitor Reading and Location*, *Sprays*, *Release Path*, and *Leak Rate*.

Verification:

- All entered values are (de)serialized from/to the project file.
- A warning appears if 100% leak rate is selected with a filtered release path.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Spent Fuel Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *Spent Fuel* from the *Source Terms Available* list. Click the *Next* button. From the *Define Spent Fuel/Spent Fuel Pool Incident* dialog, enter various values for: *Representative Operating Power*, *Number of Batches*, *Fuel Condition*, *Sprays*, *Release Path*, *Last Batch in Pool* datetime, and *Leak Rate*.

Verification:

- All entered values are (de)serialized from/to the project file.
- A warning appears if 100% leak rate is selected with a filtered release path.
- The *Last Batch in Pool* time is forced to be no later than *Shutdown/Start of Decay* on the *When* tab.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a RASCAL Project Incident

Choose the *Technical Analysis* radio button. Activate the *Define Incident* button and choose *RASCAL Project* from the *Source Terms Available* list. Click the *Next* button. From the *RASCAL Project* dialog, activate the *Read File* button to locate and select a RASCAL project file.

Verification:

- The contained *tadspecs.tmp* file is read correctly to get the case name.
- The contained *NucName.tmp* file is read correctly to get the list of isotopes.
- The contained *STC_Rel.tmp* file is read correctly to get release durations and activity values for each isotope at each release time.
- The isotope activities and durations for each release are correctly (de)serialized from/to the project file.

Test: Define a Confinement Bypass Plant Conditions Incident

In the *Where* tab, choose a facility for which the Confinement Bypass plant condition is available, such as Hartlepool-1 in Great Britain (an AGR). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Bypass of Confinement* from the *Plant Conditions Available* list. From the *Bypass of Confinement* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Confinement Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Confinement Bypass plant condition is available, such as Novovoronezh-3 in Russia (a VVER-440/230). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Confinement*

Leakage/Failure from the *Plant Conditions Available* list. From the *Confinement Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Containment Bypass Plant Conditions Incident

In the *Where* tab, choose a facility for which the Containment Bypass plant condition is available, such as Hamaoka-3 in Japan (a BWR Mk-1). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Bypass of Containment* from the *Plant Conditions Available* list. From the *Bypass of Containment* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Containment Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Containment Leakage plant condition is available, such as Beloyarsky-3(BN-600) in Russia (an LMFBR, requires a custom inventory file). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Containment Leakage/Failure* from the *Plant Conditions Available* list. From the *Containment Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Dry Well Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Dry Well Leakage plant condition is available, such as Hamaoka-3 in Japan (a BWR Mk-1). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Dry Well Leakage/Failure* from the *Plant Conditions Available* list. From the *Dry Well Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, *Sprays*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define an Ice Condenser Containment Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Ice Condenser Containment plant condition is available, such as Watts Bar-1 in the United States (a PWR). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Ice Condenser Containment Leakage/Failure* from the *Plant Conditions Available* list. From the *Ice Condenser Containment Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, *Sprays*, *Fans*, *Ice Bed Condition Before Core Damage*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define an Large Dry Subcontainment Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Large Dry Subcontainment Leakage plant condition is available, such as Beaver Valley-1 in the United States (a PWR). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Large, Dry, or Subatmospheric Containment Leakage/Failure* from the *Plant Conditions Available* list. From the *Large, Dry, or Subatmospheric Containment Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, *Sprays*, and *Leak Rate*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define an Prompt Critical Power Excursion Plant Conditions Incident

In the *Where* tab, choose a facility for which the Prompt Critical Power Excursion plant condition is available, such as Bilibino Unit A in Russia (an RBMK). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Prompt Critical Power Excursion* from the *Plant Conditions Available* list. From the *Prompt Critical Power Excursion* dialog, enter or select various values for *Representative Operating Power*, and *Core Involvement*.

Verification:

- Values entered or selected are (de)serialized from/to the project file.

Test: Define an Reprocessing Facility Plant Conditions Incident

In the *Where* tab, choose a facility for which the Reprocessing Facility plant condition is available, such as Savannah River in the United States. On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Reprocessing Facility Plant Conditions* from the *Plant Conditions Available* list. From the *Reprocessing Facility Plant Conditions* dialog, enter or select various values for *Plant Throughput*, *Release Rate*, and the values in the *Release Fraction by Component* group.

Representative Operating Power, and *Core Involvement*.
Verification:

- Values entered or selected are (de)serialized from/to the project file.

Test: Define an Steam Generator Tube Rupture Plant Conditions Incident

In the *Where* tab, choose a facility for which the Steam Tube Rupture plant condition is available, such as Dungeness B-1 in Great Britain (an AGR). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Steam Generator Tube Rupture (Coolant)* from the *Plant Conditions Available* list. From the *Steam Generator Rube Rupture (Coolant)* dialog, enter or select various values for *Coolant Concentration*, *Steam Generator Conditions*, *Release is From*, and *Release Rate*.
Verification:

- Values entered or selected are (de)serialized from/to the project file.

Test: Define an Subatmospheric Confinement Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Subatmospheric Confinement Leakage/Failure plant condition is available, such as Rovno-1 in the Ukraine (VVER-440/213). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Subatmospheric Confinement Leakage/Failure* from the *Plant Conditions Available* list. From the *Subatmospheric Confinement Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Pool Suppression System*, and *Leak Rate*.
Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: Define a Wet Well Leakage/Failure Plant Conditions Incident

In the *Where* tab, choose a facility for which the Wet Well Leakage plant condition is available, such as Hamaoka-3 in Japan (a BWR Mk-1). On the *What* tab, choose the *Technical Analysis* radio button. Activate the *Define Incident* button, choose *Plant Conditions* from the *Source Terms Available* list, activate the *Next* button, and then choose *Wet Well Leakage/Failure* from the *Plant Conditions Available* list. From the *Wet Well Leakage/Failure* dialog, enter or select various values for *Representative Operating Power*, *Core Condition*, *Release Path*, *Wet Well*, and *Leak Rate*.
Verification:

- Values entered or selected are (de)serialized from/to the project file.
- The release duration is correctly calculated from the specified leak rate.

Test: View Source Term

Activate the *View Source Term* button to generate and display an HTML report giving details of the source term as currently defined.

Verification:

- Source term HTML report generated for all source models.

5.4 WHEN TAB

The *When* tab in the NFAC GUI is relatively complex compared to other incident source models. Refer to Section 3.4 ModelTimesTest for a discription of the six event times possible in NFAC incidents. Which of the event times are visible and/or editable for a particular NFAC source model depends on that model and which of the event times are defined or computed based on the model definition. Further, the new *Advanced Mode* toggle button allows the user to override these settings and enter explicit times for all events, possibly redefining the model in the process. The interactive tests examine each NFAC source model to verify the proper times are visible and editable and are correctly computed if appropriate. The *Reset Times* button is activated to ensure the default times are applied.

Test: Moderate Model Times Edit

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, uneditable, calculated correctly
End of Release:	visible, uneditable, calculated correctly
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Severe Model Times Edit

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, uneditable, calculated correctly
End of Release:	visible, uneditable, calculated correctly
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Analyst Mix Model Times Edit

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Containment Monitor Reading Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	visible, editable
Release to Environment:	visible, editable
End of Release:	visible, editable[1]
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

1 Time can be manually set less than computed end time based on leak rate

Test: Isotopic Concentrations Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Isotopic Release Rates Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Percent Inventory Model Times Edit

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, uneditable, calculated correctly
End of Release:	visible, uneditable, calculated correctly
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Containment Bypass Plant Conditions Model Times Edit

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable[1]
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

1 Time can be manually set less than computed end time based on leak rate

Test: Prompt Critical Power Excursion Plant Conditions Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Reprocessing Facility Plant Conditions Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Confinement Bypass Plant Conditions Model Times Edit

Times management for this source model is shared by all other Plant Conditions models except for the three identified in the preceeding tests.

Verification:

Shutdown/Start of Decay:	visible, editable
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, editable[1]
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

1 Time can be manually set less than computed end time based on leak rate

Test: Spent Fuel Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	visible, editable
Release to Environment:	visible, editable
End of Release:	visible, editable[1]
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

1 Time can be manually set less than computed end time based on leak rate

Test: RASCAL Project Model Times Edit

Verification:

Shutdown/Start of Decay:	invisible
Release to Containment:	invisible
Release to Environment:	visible, editable
End of Release:	visible, uneditable, calculated correctly
End of Dispersion:	visible, editable
End of Exposure:	visible, editable

After reset, times set correctly

Test: Detect Event Times Out of Chronological Order

After modification of editable time fields and activation of the *OK* button for the *Nuclear Facility Incident Edit* dialog, if times are not in chronological order, a ”Times are not in chronological order” error dialog is displayed, and the model definition remains unchanged.

Verification:

- Out of order times are detected, a warning dialog is displayed, and the model definition is not updated.

Test: Apply Time Field Changes for All Events

Enter a field value and activate the corresponding button in the *Change Field for All Events* group.

Verification:

- Month (*MM*) fields are updated.
- Day (*DD*) fields are updated.
- Year fields are updated.
- Hours (*hh*) fields are updated.
- Minutes (*mm*) fields are updated.
- Seconds (*ss*) fields are updated.
- Calculated fields are updated correctly.

Test: Override Times in Advanced Mode

Activate the *Advanced Mode* toggle button to override editability of fields.

Verification:

- All visible fields are editable.
- Chronological order checks are still applied.

5.5 NOTES TAB

Test: Edit Notes

Verification:

- Notes are (de)serialized from/to the project file.
- Notes can be modified (changed, deleted, augmented).

6. RESULTS

6.1 AUTOMATED TEST RESULTS

NfacJUnit4Adapter consolidates all test results into a single report. The report is listed below.

```
----------------------------------------------------------------------
2015-12-02T17:03:35-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.ExternalFileFinderTest
Tests run: 4, Failures: 0, Errors: 0, Time elapsed: 2.480 sec

Test case: testNfacImpl(mil.dtra.hpac.models.nfac.test.junit.ExternalFileFinderTest)
took 1.123 sec
Passed

Test case: testFileReleases(mil.dtra.hpac.models.nfac.test.junit.ExternalFileFinderTest)
took 0.874 sec
Passed

Test case: testProject(mil.dtra.hpac.models.nfac.test.junit.ExternalFileFinderTest)
took 0.265 sec
Passed
```

```
Test case: testAllIncidents(mil.dtra.hpac.models.nfac.test.junit.ExternalFileFinderTest)
took 0.218 sec
Passed

------------------------------------------------------------------------
2015-12-02T17:03:37-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest
Tests run: 42, Failures: 0, Errors: 0, Time elapsed: 13589.501 sec

Test case: testPercentInventoryNoShutdownAsIs(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 301.330 sec
Passed

Test case: testPCIceCondenserContainmentCoreDamage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 51.646 sec
Passed

Test case: testPIRSubReleasesAverage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 125.219 sec
Passed

Test case: testPCContainmentLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 29.626 sec
Passed

Test case: testPCDryWellLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 11.650 sec
Passed

Test case: testPCLargeDrySubContainmentLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 26.088 sec
Passed

Test case: test10682GroupsNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 631.498 sec
Passed

Test case: testAnalystMix(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 15.569 sec
Passed

Test case: testPCSteamTubeRuptureGapRelease(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 18.596 sec
Passed

Test case: testPercentInventoryWithShutdownAsIs(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 311.171 sec
Passed

Test case: testPCIceCondenserContainmentGapRelease(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 43.976 sec
Passed

Test case: test11653(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 15.669 sec
Passed

Test case: test10682GroupsExplicit(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 4524.318 sec
Passed

Test case: testPercentInventoryNoShutdownForced(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 298.039 sec
Passed
```

```
Test case: testSpentFuel(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 15.746 sec
Passed

Test case: testMultiFive(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 166.247 sec
Passed

Test case: testPCSubatmosphericConfinementLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 23.412 sec
Passed

Test case: testContainmentMonitorReading(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 52.711 sec
Passed

Test case: test10682SimpleNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 77.965 sec
Passed

Test case: testPCContainmentByPass(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 12.168 sec
Passed

Test case: testPCPromptCriticalPowerExcursion(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 22.519 sec
Passed

Test case: testPCSodiumWaterReaction(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 21.731 sec
Passed

Test case: testPCConfinementByPass(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 44.606 sec
Passed

Test case: testModerate(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 109.682 sec
Passed

Test case: test10682SimpleExplicit(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 375.591 sec
Passed

Test case: testIsotopicConcentrations(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 9.498 sec
Passed

Test case: test10682UnsupportedGroupsNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 58.758 sec
Passed

Test case: testPCWetWellLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 11.558 sec
Passed

Test case: testPCConfinementLeakage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 23.711 sec
Passed

Test case: test10682RascalGroupsNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 680.833 sec
Passed

Test case: testPIRSubReleasesNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
```

```
took 108.681 sec
Passed

Test case: testPCReproFacility(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 11.597 sec
Passed

Test case: testRascalProject(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 336.176 sec
Passed

Test case: test10682VverGroupsNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 1263.811 sec
Passed

Test case: testPercentInventoryWithShutdownForced(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 313.360 sec
Passed

Test case: testPCIceCondenserContainmentMeltThrough(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 45.884 sec
Passed

Test case: testIsotopicReleaseRates(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 9.462 sec
Passed

Test case: testSevere(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 43.617 sec
Passed

Test case: testPCSteamTubeRuptureCoreDamage(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 20.033 sec
Passed

Test case: test10682RascalSimpleNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 83.051 sec
Passed

Test case: test10682PwrGroupsNone(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 242.527 sec
Passed

Test case: testPIRSubReleasesExplicit(mil.dtra.hpac.models.nfac.test.junit.ModelDefsTest)
took 3000.156 sec
Passed

-----------------------------------------------------------------------
2015-12-02T20:49:52-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest
Tests run: 8, Failures: 0, Errors: 0, Time elapsed: 0.453 sec

Test case: testAnalystMix(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.047 sec
Passed

Test case: testSpentFuel(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.031 sec
Passed

Test case: testPercentInventoryShutdown(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.047 sec
Passed

Test case: testPCContainmentByPass(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
```

```
took 0.031 sec
Passed

Test case: testPercentInventoryNoShutdown(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.032 sec
Passed

Test case: testIsotopicConcentrations(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.046 sec
Passed

Test case: testRascalProject(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.188 sec
Passed

Test case: testIsotopicReleaseRates(mil.dtra.hpac.models.nfac.test.junit.ModelTimesTest)
took 0.031 sec
Passed

------------------------------------------------------------------------
2015-12-02T20:49:52-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest
Tests run: 7, Failures: 0, Errors: 0, Time elapsed: 0.015 sec

Test case: testReadCounts(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

Test case: testBWRCounts(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

Test case: testWriteAndRead(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.015 sec
Passed

Test case: testTypeMap(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

Test case: testPWRCounts(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

Test case: testBWRValues(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

Test case: testPWRValues(mil.dtra.hpac.models.nfac.test.junit.ParticleGroupsMgrTest)
took 0.000 sec
Passed

------------------------------------------------------------------------
2015-12-02T20:49:52-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest
Tests run: 25, Failures: 0, Errors: 0, Time elapsed: 0.000 sec

Test case: testHTGRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testRBMKSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed
```

```
Test case: testPHWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testHWGCRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testGCRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testHWLWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testLWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testLGRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testVVER0SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testFBRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testLMFBRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testVVER1SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testVVER2SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testVVER3SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testAGRCompatibility(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testAGRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testBWR0SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testResearchSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testBWR1SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
```

```
took 0.000 sec
Passed

Test case: testBWR2SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testReproSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testPWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testBWR3SourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testSGHWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

Test case: testGCHWRSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest)
took 0.000 sec
Passed

------------------------------------------------------------------------
2015-12-02T20:49:52-05
Test suite: mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest2
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.000 sec

Test case: testHWRESRCSourceTerms(mil.dtra.hpac.models.nfac.test.junit.SourceTermTableTest2)
took 0.000 sec
Passed
```

6.2 INTERACTIVE TEST RESULTS

Interactive tests results are as follows.

Test	**Result**
Set Incident Name	Success
Select default facility location	Success
Enter explicit facility location	Success
Define facility location by dragging the incident Icon	Success
Select the default inventory file	Success
Select a custom inventory file	Success
Use the ORIGEN File Import Utility to create a custom inventory file	Success
Enter an explicit operating power	Success
Select a moderate incident	Success
Select a severe incident	Success
Define an Isotopic Release Rates incident	Success
Define an Isotopic Concentrations incident	Success
Define a Percent Inventory incident	Success
Define an Analyst Mix incident	Success
Define a Containment Monitor Reading incident	Success
Define a RASCAL Project incident	Success
Define a Confinement Bypass Plant Conditions incident	Success
Define a Confinement Leakage/Failure Plant Conditions incident	Success
Define a Containment Bypass Plant Conditions incident	Success
Define a Containment Leakage/Failure Plant Conditions incident	Success
Define a Dry Well Leakage/Failure Plant Conditions incident	Success
Define an Ice Condenser Containment Leakage/Failure Plant Conditions incident	Success
Define a Large Dry Subcontainmet Leakage/Failure Plant Conditions incident	Success
Define a Prompt Critical Power Excursion Plant Condition incident	Success
Define a Reprocessing Facility Plant Condition incident	Success
Define a Steam Generator Tube Rupture Plant Condition incident	Success
Define a Subatmospheric Confinement Leakage/Failure Plant Conditions incident	Success
Define a Wet Well Leakage/Failure Plant Conditions incident	Success
View source term	Success
Moderate model times edit	Success
Severe model times edit	Success
Analyst Mix model times edit	Success
Containment Monitor Reading model times edit	Success
Isotopic Concentrations model times edit	Success
Isotopic Release Rates model times edit	Success
Percent Inventory model times edit	Success
Containment Bypass Plant Conditions model Times edit	Success
Prompt Critical Power Excursion Plant Conditions model times edit	Success
Reprocessing Facility Plant Conditions model times edit	Success
Confinement Bypass Plant Conditions model times edit	Success
RASCAL Project model times edit	Success
Spent Fuel model times edit	Success
Detect event times out of chronological order	Success
Apply time field changes for all events	Success
Override times in advanced mode	Success
Edit notes	Success

References

1. R.I. Sykes, C.P. Cerasoli, and D.S. Henn. The representation of dynamic flow effects in a lagrangian puff dispersion model. *J. Haz. Mat.*, 64:223–247, 1999.

2. R.I. Sykes and R.S. Gabruk. A second-order closure model for the effect of averaging time on turbulent plume dispersion. *J. Appl. Met.*, 36:165–184, 1997.

A. NFACJUNIT4ADAPTER SOURCE LISTING

```
// $Id$
//------------------------------------------------------------------------
//      NAME:           NfacJUnit4Adapter.java                           -
//      HISTORY:                                                         -
//              2015-10-06      leerw@ornl.gov
//        SPCR 14517.
//              2015-07-15      leerw@ornl.gov
//        Added timestamp to start of test suite in report.
//              2015-03-04      leerw@ornl.gov
//        Added RunReporter and reporting to a file.
//              2014-12-30      leerw@ornl.gov
//        Calling Bundle.loadClass().
//              2014-12-29      leerw@ornl.gov
//        Saving BundleContext for shutting down after run.
//              2014-11-13      leerw@ornl.gov
//------------------------------------------------------------------------
package mil.dtra.hpac.models.nfac.test.junit;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.TreeSet;

import java.util.logging.Level;
import java.util.logging.Logger;

import org.junit.runner.Description;
import org.junit.runner.JUnitCore;
import org.junit.runner.Request;
import org.junit.runner.Result;
import org.junit.runner.Runner;

import org.junit.runner.manipulation.Filter;

import org.junit.runner.notification.Failure;
import org.junit.runner.notification.RunListener;

import org.osgi.framework.Bundle;
import org.osgi.framework.BundleContext;
//import org.osgi.framework.FrameworkUtil;


//------------------------------------------------------------------------
//      CLASS:          NfacJUnit4Adapter                                -
/*************************************************************************
* Tests are defined in a comma-separated list of test class names with
* an optional colon-separated list of methods following each test class
* name.
************************************************************************/
public class
NfacJUnit4Adapter
    implements Runnable
  {
                // Constants
```

```
              //

public final static
  String              BASE_NAME = "nfac.test.junit.NfacJUnit4Adapter";

public final static
  String              PROP_report = BASE_NAME + ".report";

public final static
  String              PROP_tests = BASE_NAME + ".tests";


              // Class Attributes
              //

protected final static
  Logger              fLogger_ =
      Logger.getLogger( NfacJUnit4Adapter.class.getPackage().getName() );


              // Object Attributes
              //

private
  BundleContext       fBundleContext;

private
  String              fReportFilename;

private
  ArrayList<Runner>   fRunnerList;


              // Object Methods
              //


//-----------------------------------------------------------------------
//    METHOD:         NfacJUnit4Adapter()                               -
/************************************************************************
***********************************************************************/
public
NfacJUnit4Adapter()
  {
  fLogger_.log( Level.INFO, "instantiated" );
  fRunnerList = new ArrayList<>( 64 );
  } // NfacJUnit4Adapter


//-----------------------------------------------------------------------
//    METHOD:         activate()                                        -
/************************************************************************
* @return             object reference
***********************************************************************/
//@Activate
public void
activate( BundleContext context )
  {
  fLogger_.log( Level.INFO, "activated" );

  setBundleContext( context );
  new Thread( this ).start();

  System.err.println( "[NfacJUnit4Adapter.activate] run thread launched" );
  fLogger_.log( Level.INFO, "run thread launched" );
  } // activate


//-------------------------------------------------------------------
//    METHOD:         deactivate()                                  -
/*******************************************************************
* @return             object reference
*******************************************************************/
//@Deactivate
public void
deactivate( BundleContext context )
  {
  fLogger_.log( Level.INFO, "deactivated" );
  setBundleContext( null );
  } // deactivate


//-------------------------------------------------------------------
//    METHOD:         getBundleContext()                            -
/*******************************************************************
* @return             object reference
*******************************************************************/
public BundleContext
getBundleContext()
  {
  return  fBundleContext;
  } // getBundleContext


//-------------------------------------------------------------------
//    METHOD:         getReportFilename()                           -
/*******************************************************************
*******************************************************************/
public String
getReportFilename()
  {
  return  fReportFilename;
  } // getReportFilename


//-------------------------------------------------------------------
//    METHOD:         getRunnerList()                               -
/*******************************************************************
* @return             object reference
*******************************************************************/
protected List<Runner>
getRunnerList()
  {
  return  fRunnerList;
  } // getRunnerList


//-------------------------------------------------------------------
//    METHOD:         loadClass()                                   -
/*******************************************************************
*******************************************************************/
public Class<?>
loadClass( String class_name )
    throws ClassNotFoundException
  {
  getLogger().info( "class_name=" + class_name );

              // Assert on Bundle
              //
  Bundle  bundle;
  if (
      getBundleContext() == null ||
      (bundle = getBundleContext().getBundle()) == null
      )
    throw  new IllegalStateException( "BundleContext has not been set" );

  return  bundle.loadClass( class_name );
  } // loadClass


//----------------------------------------------------------------------
//    METHOD:         run()                                            -
/***********************************************************************
***********************************************************************/
public void
run()
  {
            // Get Output Filename
            //
  setReportFilename( System.getProperty( PROP_report ) );

            // Get List of Tests/Methods to Run
            //
  getRunnerList().clear();
  String  defs_str = System.getProperty( PROP_tests, "" );

  for ( String item : defs_str.split( "," ) )
    {
                    // Remove Trailing Colons, Then Split
                    //
    item = item.replaceAll( ":*$", "" );
    String[]  tokens = item.split( ":" );

                    // Class Name Cannot be Blank
                    //
    if ( tokens[ 0 ].length() > 0 )
      {
      try
        {
        getLogger().info( "loading class: " + tokens[ 0 ] );
        Class<?>  test_class = loadClass( tokens[ 0 ] );
        Request  request;

                            // Methods Specified?
                            //
        if ( tokens.length > 1 )
          {
          String[]  methods = Arrays.copyOfRange( tokens, 1, tokens.length );
          getLogger().info( "methods: " + Arrays.toString( methods ) );
          Filter  filter = new MethodsFilter( test_class, methods );
          request = Request.aClass( test_class ).filterWith( filter );
          } // if methods specified

        else
          request = Request.aClass( test_class );

        getLogger().info( "running test(s)" );
        getRunnerList().add( request.getRunner() );
        } // try

      catch ( ClassNotFoundException ex )
        {
        getLogger().log(
            Level.WARNING,
            String.format( "Error loading class ’%s’", tokens[ 0 ] ),
            ex
            );
        }
      } // if class specified
    } // for each class item

              // Anything to Run?
              //
  if ( getRunnerList().size() > 0 )
    {
    JUnitCore  junit = new JUnitCore();
    RunReporter  reporter = new RunReporter( getReportFilename() );

    junit.addListener( new RunReporter( getReportFilename() ) );
    try
      {
      for ( Runner runner : getRunnerList() )
        runOne( junit, runner );
      }
    finally
      {
      reporter.close();
      }
    } // if runners to run

              // Quit
              //
  Bundle  sys_bundle;
  if (
      getBundleContext() != null &&
      (sys_bundle = getBundleContext().getBundle( 0L )) != null
      )
    {
    System.err.println( "[NfacJUnit4Adapter.run] stopping bundle 0" );
    fLogger_.info( "stopping bundle 0" );

    try { sys_bundle.stop(); }
    catch ( Exception ex )
      {
      String  message = "Error stopping system bundle";
      System.err.println( "[NfacJUnit4Adapter.run] " + message );
      fLogger_.log( Level.WARNING, message, ex );
      }
    }
  } // run


//----------------------------------------------------------------------
//    METHOD:         runOne()                                          -
/***********************************************************************
***********************************************************************/
protected void
runOne( JUnitCore junit, Runner runner )
  {
  Result  result = junit.run( runner );
  System.out.printf(
      "%n%n%s  (%.3fs)%nTests: %3d  Failures: %3d  %s%n",
      runner.getDescription().getClassName(),
      result.getRunTime() / 1000.,
      result.getRunCount(),
      result.getFailureCount(),
      result.wasSuccessful() ? "OK" : "Failed"
      //result.getFailureCount() == 0 ? "OK" : "Failed"
      );

  for ( Failure failure : result.getFailures() )
    System.out.printf( "%n%s%n", failure );

  System.out.println();
  } // runOne


//-------------------------------------------------------------------
//    METHOD:         runOne_0()                                    -
/********************************************************************
*******************************************************************/
protected void
runOne_0( JUnitCore junit, Runner runner )
  {
  Result  result = junit.run( runner );

  PrintStream  report = System.out;
  if ( getReportFilename() != null )
    {
    try
      {
      report = new PrintStream(
          new FileOutputStream( getReportFilename(), true ),
          true
          );
      }
    catch ( IOException ex )
      {
      report = System.out;
      }
    } // if report filename specified

  try
    {
    report.printf(
        "%n%n%s  (%.3fs)%nTests: %3d  Failures: %3d  %s%n",
        runner.getDescription().getClassName(),
        result.getRunTime() / 1000.,
        result.getRunCount(),
        result.getFailureCount(),
        result.wasSuccessful() ? "OK" : "Failed"
        //result.getFailureCount() == 0 ? "OK" : "Failed"
        );

    for ( Failure failure : result.getFailures() )
      report.printf( "%n%s%n", failure );

    report.println();
    } // try

  finally
    {
    if ( report != System.out )
      report.close();
    }
  } // runOne_0


//-------------------------------------------------------------------
//    METHOD:         setBundleContext()                            -
/********************************************************************
* @param  value       object reference to store
*******************************************************************/
protected void
setBundleContext( BundleContext value )
```

```
  {
  fBundleContext = value;
  } // setBundleContext


//----------------------------------------------------------------------
//    METHOD:         setReportFilename()                              -
/***********************************************************************
***********************************************************************/
protected void
setReportFilename( String value )
  {
  fReportFilename = value;
  } // setReportFilename


              // Class Methods
              //


//----------------------------------------------------------------------
//    METHOD:         getLogger()                                      -
/***********************************************************************
* @return             object reference
***********************************************************************/
public static Logger
getLogger()
  {
  return  fLogger_;
  } // getLogger


//----------------------------------------------------------------------
//    METHOD:         loadClass_pojo()                                 -
/***********************************************************************
***********************************************************************/
public static Class<?>
loadClass_pojo( String class_name )
    throws ClassNotFoundException
  {
  getLogger().info( "class_name=" + class_name );

  try
    {
    getLogger().info( "trying class.classLoader" );
    return  NfacJUnit4Adapter.class.getClassLoader().loadClass( class_name );
    }

  catch ( ClassNotFoundException ex )
    {
    try
      {
      getLogger().info( "trying thread.contextClassLoader" );
      return
      Thread.currentThread().getContextClassLoader().loadClass( class_name );
      }

    catch ( ClassNotFoundException ex2 )
      {
      getLogger().info( "trying Class.forName()" );
      return  Class.forName( class_name );
      }
    }
  } // loadClass_pojo

```

```

//------------------------------------------------------------------
//    METHOD:         main()                                        -
/*******************************************************************
*******************************************************************/
public static void
main( String[] argv )
  {
  try
    {
    new NfacJUnit4Adapter().run();
    }

  catch ( Exception ex )
    {
    ex.printStackTrace( System.err );
    }
  } // main


              // Inner Classes
              //


//------------------------------------------------------------------
//    CLASS:          MethodsFilter                                 -
/*******************************************************************
*******************************************************************/
protected final static class
MethodsFilter
    extends Filter
  {
  private
    TreeSet<String>   fMethodNames;

  private
    Class<?>          fTestClass;


  //----------------------------------------------------------------
  //  METHOD:         MethodsFilter.MethodsFilter()                 -
  /*****************************************************************
  *****************************************************************/
  public
  MethodsFilter( Class<?> test_class, String[] method_names )
    {
    fTestClass = test_class;

    fMethodNames = new TreeSet<String>();
    if ( method_names != null && method_names.length > 0 )
      {
      for ( String name : method_names )
        fMethodNames.add( name );
      }
    } // MethodsFilter


  //----------------------------------------------------------------
  //  METHOD:         MethodsFilter.describe()                      -
  /*****************************************************************
  *****************************************************************/
  public String
  describe()
    {
    StringBuilder  buffer = new StringBuilder( 256 );
```

```
      buffer.append( fTestClass.getName() );

      if ( fMethodNames.size() > 0 )
        {
/*Java8
        buffer.
            append( ":" ).
            append( String.join( ":", fMethodNames ) );
*/
        buffer.append( ":" );
        for ( String name : fMethodNames )
          buffer.append( ":" ).append( name );
        }

      return  buffer.toString();
      } // describe


    //-------------------------------------------------------------------
    //  METHOD:         MethodsFilter.shouldRun()                      -
    /********************************************************************
    *******************************************************************/
    public boolean
    shouldRun( Description descr )
      {
      boolean  result = false;

      if ( descr.isTest() )
        {
        result =
            fTestClass.getName().equals( descr.getClassName() ) &&
            descr.getMethodName() != null &&
            fMethodNames.contains( descr.getMethodName() );
        }

      else
        {
        for ( Description child : descr.getChildren() )
          {
          if ( shouldRun( child ) )
            {
            result = true;
            break;
            }
          }
        } // else not a test

      return  result;
      } // shouldRun
    } // MethodsFilter


  //---------------------------------------------------------------------
  //    CLASS:          RunReporter()                                  -
  /**********************************************************************
  *********************************************************************/
  public static class
  RunReporter
      extends RunListener
    {
    private
      StringBuilder     fBuffer;

    private transient
      SimpleDateFormat  fDateFormat;
```

```

private
  List<Failure>     fFailures;

private
  PrintStream       fPrint;

private
  long              fTestStart;


//----------------------------------------------------------------------
//  METHOD:         RunReporter.RunReporter()                          -
/*********************************************************************
*********************************************************************/
public
RunReporter( String report_filename )
  {
  fBuffer = new StringBuilder( 32768 );
  fDateFormat = new SimpleDateFormat( "yyyy-MM-dd'T'HH:mm:ssX" );
  fFailures = new ArrayList<>( 128 );
  fPrint = null;

  if ( report_filename != null )
    {
    try
      {
      fPrint = new PrintStream(
          new FileOutputStream( report_filename, true ),
          true
          );
      }
    catch ( IOException ex ) {}
    } // if report filename specified

  if ( fPrint == null )
    fPrint = System.out;
  } // RunReporter


//----------------------------------------------------------------------
//  METHOD:         RunReporter.close()                                -
/*********************************************************************
* @return           object reference
*********************************************************************/
public void
close()
  {
  fPrint.close();
  } // close


//----------------------------------------------------------------------
//  METHOD:         RunReporter.finalize()                             -
/*********************************************************************
* @return           object reference
*********************************************************************/
@Override
protected void
finalize()
  {
  close();
  } // finalize


```

```
//-------------------------------------------------------------------
//  METHOD:         RunReporter.getBuffer()                         -
/*******************************************************************
* @return           object reference
*******************************************************************/
protected StringBuilder
getBuffer()
  {
  return  fBuffer;
  } // getBuffer


//-------------------------------------------------------------------
//  METHOD:         RunReporter.getDateFormat()                     -
/*******************************************************************
* @return           object reference
*******************************************************************/
protected DateFormat
getDateFormat()
  {
  return  fDateFormat;
  } // getDateFormat


//-------------------------------------------------------------------
//  METHOD:         RunReporter.getFailures()                       -
/*******************************************************************
* @return           object reference
*******************************************************************/
protected List<Failure>
getFailures()
  {
  return  fFailures;
  } // getFailures


//-------------------------------------------------------------------
//  METHOD:         RunReporter.getPrint()                          -
/*******************************************************************
* @return           object reference
*******************************************************************/
public PrintStream
getPrint()
  {
  return  fPrint;
  } // getPrint


//-------------------------------------------------------------------
//  METHOD:         RunReporter.getTestStart()                      -
/*******************************************************************
*******************************************************************/
public long
getTestStart()
  {
  return  fTestStart;
  } // getTestStart


//-------------------------------------------------------------------
//  METHOD:         RunReporter.setTestStart()                      -
/*******************************************************************
*******************************************************************/
protected void
setTestStart( long value )
```

```
  {
  fTestStart = value;
  } // setTestStart


//-------------------------------------------------------------------
//  METHOD:         RunReporter.testAssumptionFailure()             -
/********************************************************************
*******************************************************************/
@Override
public void
testAssumptionFailure( Failure failure )
  {
  testFailure( failure );
  } // testAssumptionFailure


//-------------------------------------------------------------------
//  METHOD:         RunReporter.testFailure()                       -
/********************************************************************
*******************************************************************/
@Override
public void
testFailure( Failure failure )
  {
  getFailures().add( failure );
  //report.printf( "%n%s%n", failure );
  } // testFailure


//-------------------------------------------------------------------
//  METHOD:         RunReporter.testFinished()                      -
/********************************************************************
*******************************************************************/
@Override
public void
testFinished( Description descr )
  {
  long  now = System.currentTimeMillis();
  getBuffer().append( String.format(
      "%nTest case: %s took %.3f sec%n\t%s%n",
      descr.getDisplayName(),
      (now - getTestStart()) / 1000.,
      getFailures().size() > 0 ? "FAILED" : "Passed"
      ) );

  for ( Failure failure : getFailures() )
    getBuffer().append( String.format( "  %s%n", failure ) );
  } // testFinished


//-------------------------------------------------------------------
//  METHOD:         RunReporter.testIgnored()                       -
/********************************************************************
*******************************************************************/
@Override
public void
testIgnored( Description descr )
  {
  } // testIgnored


//-------------------------------------------------------------------
//  METHOD:         RunReporter.testRunFinished()                   -
/********************************************************************
```

```
    *****************************************************************/
    @Override
    public void
    testRunFinished( Result result )
      {
      fPrint.printf(
          "Tests run: %d, Failures: %d, Errors: %d, Time elapsed: %.3f sec%n",
          result.getRunCount(),
          result.getFailureCount(),
          0, //result.getRunCount() - result.getFailureCount(),
          result.getRunTime() / 1000.
          );

      fPrint.println( getBuffer().toString() );
      } // testRunFinished


    //--------------------------------------------------------------------
    //  METHOD:         RunReporter.testRunStarted()                     -
    /*******************************************************************
    *****************************************************************/
    @Override
    public void
    testRunStarted( Description descr )
      {
      getBuffer().setLength( 0 );
      getFailures().clear();

      fPrint.println( "------------------------------------------------------------------------" );
      fPrint.printf(
          "%s%nTest suite: %s%n",
          getDateFormat().format( new Date() ),
          descr.getDisplayName()
          );
      fPrint.flush();
      } // testRunStarted


    //--------------------------------------------------------------------
    //  METHOD:         RunReporter.testStarted()                        -
    /*******************************************************************
    *****************************************************************/
    @Override
    public void
    testStarted( Description descr )
      {
      getFailures().clear();
      setTestStart( System.currentTimeMillis() );
      //report.printf( "Starting %s%n", descr.getDisplayName() );
      } // testStarted
    } // RunReporter
  } // NfacJUnit4Adapter
```

B. NFACLOG SOURCE LISTING

```
// $Id$
//---------------------------------------------------------------------
//      NAME:           NfacLog.java                                  -
//      HISTORY:                                                      -
//              2015-01-12      leerw@ornl.gov
//        Eight chars in material name, matching decay time to ATIs.
//              2014-03-03      leerw@ornl.gov
//        New outputs.
//              2014-02-19      leerw@ornl.gov
//---------------------------------------------------------------------
package mil.dtra.hpac.models.nfac.test;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;

import java.text.DecimalFormat;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

//import java.util.logging.Level;
//import java.util.logging.Logger;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import java.util.zip.GZIPInputStream;

import mil.dtra.hpac.models.nfac.CAcomp.data.ModelTimes;

import mil.dtra.hpac.util.DataUtils;
import mil.dtra.hpac.util.Time;


//---------------------------------------------------------------------
//      CLASS:          NfacLog                                       -
/*********************************************************************
* Captures the NFAC log "FINE" level output showing inventories and
* activities.
*********************************************************************/
public class
NfacLog
  {
                // Constants
                //

  public final static
    String              BASE_NAME = "nfac.test.NfacLog";

  public final static
    Pattern             PATTERN_ati =
        Pattern.compile( "ˆactivityTableInput: name=[ˆ,]*, matID=(.*)$" );

  public final static
    Pattern             PATTERN_releaseBaseActivity =
        Pattern.compile( "ˆ\\Q++\\Erelease_base_activity, pir_ndx=([0-9]+):$" );
```

```

public final static
  Pattern             PATTERN_ws = Pattern.compile( "[\\s,]+" );

public final static
  String              PROP_debug = BASE_NAME + ".debug";


              // Object Attributes
              //

private
  ActivityInputRec[]  fActivityInputs;

private
  IsotopeRec[]        fFacilityInventory;

private
  ModelTimes          fModelTimes;

private
  ReleaseRec[]        fReleases;


              // Object Methods
              //


//----------------------------------------------------------------------
//    METHOD:         NfacLog()                                        -
/***********************************************************************
***********************************************************************/
public
NfacLog( File file )
    throws IOException
  {
  read( file );
  } // NfacLog


//----------------------------------------------------------------------
//    METHOD:         NfacLog()                                        -
/***********************************************************************
***********************************************************************/
public
NfacLog( InputStream input )
    throws IOException
  {
  read( input );
  } // NfacLog


//----------------------------------------------------------------------
//    METHOD:         equals()                                         -
/***********************************************************************
***********************************************************************/
@Override
public boolean
equals( Object that )
  {
  return  !(that instanceof NfacLog) ? false : equals( (NfacLog)that );
  } // equals


//----------------------------------------------------------------------
```

```
//    METHOD:         equals()                                      -
/*********************************************************************
*********************************************************************/
public boolean
equals( NfacLog that )
  {
  boolean  equal = false;

  if ( Boolean.getBoolean( PROP_debug ) )
    {
    boolean[]  eq = new boolean[ 4 ];

    eq[ 0 ] = DataUtils.equals( this.getModelTimes(), that.getModelTimes() );
    System.err.printf( "[NfacLog.equals] modelTimes=%b%n", eq[ 0 ] );

    eq[ 1 ] = Arrays.
        equals( this.getFacilityInventory(), that.getFacilityInventory() );
    System.err.printf( "[NfacLog.equals] facilityInventory=%b%n", eq[ 1 ] );

    eq[ 2 ] = Arrays.equals( this.getReleases(), that.getReleases() );
    System.err.printf( "[NfacLog.equals] releases=%b%n", eq[ 2 ] );

    eq[ 3 ] = Arrays.
        equals( this.getActivityInputs(), that.getActivityInputs() );
    System.err.printf( "[NfacLog.equals] activityInputs=%b%n", eq[ 3 ] );

    equal = true;
    for ( int i = 0; i < eq.length && equal; i++ )
      equal &= eq[ i ];
    }

  else
    {
    equal =
      DataUtils.equals( getModelTimes(), that.getModelTimes() ) &&
      Arrays.equals( getFacilityInventory(), that.getFacilityInventory() ) &&
      Arrays.equals( getReleases(), that.getReleases() ) &&
      Arrays.equals( getActivityInputs(), that.getActivityInputs() );
    }

  return  equal;
  } // equals


//-------------------------------------------------------------------
//    METHOD:         getActivityInputs()                           -
/*********************************************************************
* @return             array reference
*********************************************************************/
public ActivityInputRec[]
getActivityInputs()
  {
  return  fActivityInputs;
  } // getActivityInputs


//-------------------------------------------------------------------
//    METHOD:         getFacilityInventory()                        -
/*********************************************************************
* @return             array reference
*********************************************************************/
public IsotopeRec[]
getFacilityInventory()
  {
  return  fFacilityInventory;
  } // getFacilityInventory


//-------------------------------------------------------------------
//    METHOD:         getModelTimes()                               -
/*******************************************************************
* @return             object reference
*******************************************************************/
public ModelTimes
getModelTimes()
  {
  return  fModelTimes;
  } // getModelTimes


//-------------------------------------------------------------------
//    METHOD:         getReleases()                                 -
/*******************************************************************
* @return             array reference
*******************************************************************/
public ReleaseRec[]
getReleases()
  {
  return  fReleases;
  } // getReleases


//-------------------------------------------------------------------
//    METHOD:         read()                                        -
/*******************************************************************
* Calls {@link #read( InputStream )}.
*******************************************************************/
public void
read( File file )
    throws IOException
  {
  InputStream  input = new FileInputStream( file );

  if ( file.getName().endsWith( ".gz" ) )
    input = new GZIPInputStream( input );

  try
    { read( input ); }
  finally
    { input.close(); }
  } // read


//-------------------------------------------------------------------
//    METHOD:         read()                                        -
/*******************************************************************
*******************************************************************/
public void
read( InputStream input )
    throws IOException
  {
  BufferedReader  reader =
      new BufferedReader( new InputStreamReader( input ) );
  String  line, state = null;

  IsotopeRec[]  facility_inv = null;
  ArrayList<ReleaseRec>  release_recs = new ArrayList<>( 16 );
  ActivityInputRec[]  ati_recs = null;

              // Outer Read Loop
```

```
                //
    while ( (line = reader.readLine()) != null )
      {
                        // Model Times
                        //
      if ( line.startsWith( "[ModelTimes.begin]:" ) )
        setModelTimes( readModelTimes( reader ) );

                        // Facility Inventory
                        //
      else if ( line.startsWith( "+facility_inventory:" ) )
        {
        facility_inv = IsotopeRec.readArray( reader );

        float[]  decay_times = readResolvedDecayTimes( reader );

        ReleaseRec  release_rec;
        while ( (release_rec = ReleaseRec.readNext( reader )) != null )
          release_recs.add( release_rec );
        }

                        // Facility Inventory
                        //
      else if ( line.startsWith( "FINE: +activityTableInputs:" ) )
        ati_recs = ActivityInputRec.readArrayEnd( reader, null );
      else if ( line.startsWith( "activityTableInput:" ) )
        ati_recs = ActivityInputRec.readArrayEnd( reader, line );
      } // while

    setFacilityInventory( facility_inv );
    setReleases( release_recs.toArray( new ReleaseRec[ release_recs.size() ] ) );
    setActivityInputs( ati_recs );
    } // read


  //----------------------------------------------------------------------
  //    METHOD:         readModelTimes()                                 -
  /***********************************************************************
  ***********************************************************************/
  public ModelTimes
  readModelTimes( BufferedReader reader )
      throws IOException
    {
    String  line;
    Time
        start = new Time(),
        rel_cont = new Time(),
        rel_env = new Time(),
        end_rel = new Time(),
        end_disp = new Time(),
        end_exp = new Time();

    while (
        (line = reader.readLine()) != null &&
        ! line.startsWith( "[ModelTimes.end]" )
        )
      {
      int  comma_ndx = line.indexOf( "," );
      if ( comma_ndx > 0 )
        {
        line = line.substring( 0, comma_ndx ).trim();
        if ( line.startsWith( "start=" ) )
          start.valueOf( line.substring( 6 ) );
        else if ( line.startsWith( "relCont=" ) )
          rel_cont.valueOf( line.substring( 8 ) );
        else if ( line.startsWith( "relEnv=" ) )
          rel_env.valueOf( line.substring( 7 ) );
        else if ( line.startsWith( "endRel=" ) )
          end_rel.valueOf( line.substring( 7 ) );
        else if ( line.startsWith( "endDisp=" ) )
          end_disp.valueOf( line.substring( 8 ) );
        else if ( line.startsWith( "endExp=" ) )
          end_exp.valueOf( line.substring( 7 ) );
        }
      } // while

    return
        new ModelTimes( start, rel_cont, rel_env, end_rel, end_disp, end_exp );
    } // readModelTimes


  //----------------------------------------------------------------------
  //    METHOD:         readResolvedDecayTimes()                         -
  /***********************************************************************
  ***********************************************************************/
  public float[]
  readResolvedDecayTimes( BufferedReader reader )
      throws IOException
    {
    float[]  times = null;
    String  line;
    int  eq_ndx;

    while (
        (line = reader.readLine()) != null &&
        ! line.contains( "resolved decay_times=" )
        )
      ;

    if ( line != null && (eq_ndx = line.indexOf( "=" )) >= 0 )
      {
      line = line.substring( eq_ndx + 2, line.length() - 1 );
      String[]  tokens = PATTERN_ws.split( line );

      times = new float[ tokens.length ];
      for ( int i = 0; i < tokens.length; i++ )
        times[ i ] = DataUtils.parseFloat( tokens[ i ] );
      } // if

    return  times == null ? new float[ 0 ] : times;
    } // readResolvedDecayTimes


  //----------------------------------------------------------------------
  //    METHOD:         setActivityInputs()                              -
  /***********************************************************************
  * @param  value       array reference to store
  ***********************************************************************/
  public void
  setActivityInputs( ActivityInputRec[] value )
    {
    fActivityInputs = value;
    } // setActivityInputs


  //----------------------------------------------------------------------
  //    METHOD:         setFacilityInventory()                           -
  /***********************************************************************
  * @param  value       array reference to store
  ***********************************************************************/
```

```
public void
setFacilityInventory( IsotopeRec[] value )
  {
  fFacilityInventory = value;
  } // setFacilityInventory


//-----------------------------------------------------------------------
//    METHOD:         setModelTimes()                                  -
/***********************************************************************
* @param  value       object reference to store
***********************************************************************/
public void
setModelTimes( ModelTimes value )
  {
  fModelTimes = value;
  } // setModelTimes


//-----------------------------------------------------------------------
//    METHOD:         setReleases()                                    -
/***********************************************************************
* @param  value       array reference to store
***********************************************************************/
public void
setReleases( ReleaseRec[] value )
  {
  fReleases = value;
  } // setReleases


//-----------------------------------------------------------------------
//    METHOD:         write()                                          -
/***********************************************************************
***********************************************************************/
public void
write( PrintStream writer )
    throws IOException
  {
  DecimalFormat  value_fmt = new DecimalFormat( "0.0##E00" );

  if ( getFacilityInventory() != null )
    {
    writer.printf(
        "+facility_inventory:%n%ss%n",
        IsotopeRec.toHeaderString()
        );
    for ( IsotopeRec iso_rec : getFacilityInventory() )
      writer.println( iso_rec.toPrettyString( value_fmt ) );
    }

  for ( ReleaseRec release_rec : getReleases() )
    {
    writer.printf( "%n%n" );
    release_rec.write( writer );
    }

  for ( ActivityInputRec ati_rec : getActivityInputs() )
    {
    writer.printf( "%n%n" );
    ati_rec.write( writer );
    }
  } // write


```

```
            // Class Methods
            //


//---------------------------------------------------------------------
//    METHOD:         main()                                          -
/**********************************************************************
**********************************************************************/
public static void
main( String[] argv )
  {
  try
    {
    if ( argv.length < 2 )
      System.err.println( "Usage: NfacLog { -i file | file1 file2 }" );

    else if ( "-i".equals( argv[ 0 ] ) )
      new NfacLog( new File( argv[ 1 ] ) ).write( System.out );

    else
      {
      NfacLog
          one = new NfacLog( new File( argv[ 0 ] ) ),
          two = new NfacLog( new File( argv[ 1 ] ) );

      System.out.println( one.equals( two ) ? "EQUAL" : "not equal" );
      }
    } // try

  catch ( Exception ex )
    {
    ex.printStackTrace( System.err );
    }
  } // main


            // Inner Classes
            //


//---------------------------------------------------------------------
//    CLASS:          ActivityInputRec                                -
/**********************************************************************
**********************************************************************/
public static class
ActivityInputRec
    implements Comparable<ActivityInputRec>
  {
  public float        decaytime;

  public IsotopeRec[] isotopes;

  public String       material;


  //-------------------------------------------------------------------
  //  METHOD:         ActivityInputRec()                              -
  /********************************************************************
  ********************************************************************/
  public
  ActivityInputRec( String material, IsotopeRec[] isotopes, float decay_time )
    {
    this.decaytime = decay_time;
    this.isotopes = isotopes;
    this.material = material;
```

```
      } // ActivityInputRec


    //----------------------------------------------------------------------
    //  METHOD:         compareTo()                                       -
    /**********************************************************************
    **********************************************************************/
    @Override
    public int
    compareTo( ActivityInputRec that )
      {
      String
          this_name = RadFile.getMaterialIndexName( this.material ),
          that_name = RadFile.getMaterialIndexName( that.material );

      return
          this_name == null ? (that_name == null ? 0 : -1) :
          that_name == null ? 1 :
          this_name.compareTo( that_name );
      } // compareTo


    //----------------------------------------------------------------------
    //  METHOD:         equals()                                          -
    /**********************************************************************
    **********************************************************************/
    @Override
    public boolean
    equals( Object that )
      {
      boolean  equal = false;

      if ( that instanceof ActivityInputRec )
        {
        ActivityInputRec  rthat = (ActivityInputRec)that;

        equal =
            DataUtils.equals( this.material, rthat.material ) &&
            Arrays.equals( this.isotopes, rthat.isotopes );
        } // if a ReleaseRec

      return  equal;
      } // equals


    //----------------------------------------------------------------------
    //  METHOD:         write()                                           -
    /**********************************************************************
    **********************************************************************/
    public void
    write( PrintStream writer )
        throws IOException
      {
      DecimalFormat  value_fmt = new DecimalFormat( "0.0##E00" );

      writer.printf(
          "+activity_table_input name=%s decay=%g:%n%s%n",
          this.material, this.decaytime, IsotopeRec.toHeaderString()
          );
      for ( IsotopeRec iso_rec : this.isotopes )
        writer.println( iso_rec.toPrettyString( value_fmt ) );
      } // write


    //----------------------------------------------------------------------
```

```
//  METHOD:         readArray()                                     -
/*******************************************************************
*******************************************************************/
public static ActivityInputRec[]
readArray( BufferedReader reader, float[] decay_times )
    throws IOException
  {
  String  line;
  ArrayList<ActivityInputRec>  rec_list = new ArrayList<>( 16 );

  while (
      (line = reader.readLine()) != null &&
      ! line.contains( "+activityTableInputs:" )
      )
    ;

  int  decay_ndx = 0;
  while (
      (line = reader.readLine()) != null &&
      (line = line.trim()).length() > 0
      )
    {
    int  ndx = line.lastIndexOf( "=" );
    String  name = line.substring( ndx + 1 ).substring( 0, 10 );

    line = reader.readLine();
    IsotopeRec[]  isotopes = IsotopeRec.readArray( reader );

    float  decay_time =
        decay_ndx < decay_times.length ? decay_times[ decay_ndx ] : -1.f;
    rec_list.add( new ActivityInputRec( name, isotopes, decay_time ) );

    decay_ndx++;
    } // while reading blocks

  Collections.sort( rec_list );
  return  rec_list.toArray( new ActivityInputRec[ rec_list.size() ] );
  } // readArray


//-------------------------------------------------------------------
//  METHOD:         readArrayEnd()                                  -
/*******************************************************************
*******************************************************************/
public static ActivityInputRec[]
readArrayEnd( BufferedReader reader, String line )
    throws IOException
  {
  //String  line;
  ArrayList<ActivityInputRec>  rec_list = new ArrayList<>( 16 );

  if ( line == null )
    line = reader.readLine();

  while (
      //(line = reader.readLine()) != null &&
      line != null &&
      ! line.contains( "ModelDefUtils logIncidentObjects" )
      )
    {
    Matcher  m = PATTERN_ati.matcher( line );
    if ( m.matches() )
      {
      String  name = m.group( 1 ).substring( 0, 10 );
      reader.readLine();
```

```

        IsotopeRec[]  isotopes = IsotopeRec.readArray( reader );
        rec_list.add( new ActivityInputRec( name, isotopes, -1.f ) );
        }

      line = reader.readLine();
      } // outer while

    Collections.sort( rec_list );
    return  rec_list.toArray( new ActivityInputRec[ rec_list.size() ] );
    } // readArrayEnd
  } // ActivityInputRec


//------------------------------------------------------------------
//    CLASS:          IsotopeRec                                   -
/*******************************************************************
*******************************************************************/
public static class
IsotopeRec
    implements Comparable<IsotopeRec>
  {
  public double       activity;
  public int          dep;
  public int          group;
  public String       name;


  //----------------------------------------------------------------
  //  METHOD:         IsotopeRec()                                 -
  /*****************************************************************
  *****************************************************************/
  public
  IsotopeRec( String line )
    {
    valueOf( line );
    } // IsotopeRec


  //----------------------------------------------------------------
  //  METHOD:         IsotopeRec()                                 -
  /*****************************************************************
  *****************************************************************/
  public
  IsotopeRec( String name, int dep, double activity )
    {
    this( name, dep, -1, activity );
    } // IsotopeRec


  //----------------------------------------------------------------
  //  METHOD:         IsotopeRec()                                 -
  /*****************************************************************
  *****************************************************************/
  public
  IsotopeRec( String name, int dep, int group, double activity )
    {
    this.name = name;
    this.dep = dep;
    this.group = group;
    this.activity = activity;
    } // IsotopeRec


  //----------------------------------------------------------------
```

```
//  METHOD:         compareTo()                                    -
/*****************************************************************
*****************************************************************/
@Override
public int
compareTo( IsotopeRec that )
  {
  return  this.name.compareTo( that.name );
  } // compareTo


//----------------------------------------------------------------
//  METHOD:         equals()                                       -
/*****************************************************************
*****************************************************************/
@Override
public boolean
equals( Object that )
  {
  boolean  equal = false;

  if ( that instanceof IsotopeRec )
    {
    IsotopeRec  rthat = (IsotopeRec)that;
    equal =
        DataUtils.equals( this.name, rthat.name ) &&
        this.dep == rthat.dep &&
        this.group == rthat.group &&
        this.activity == rthat.activity;
    }

  return  equal;
  } // equals


//----------------------------------------------------------------
//  METHOD:         toPrettyString()                               -
/*****************************************************************
*****************************************************************/
public String
toPrettyString( DecimalFormat value_fmt )
  {
  return
  String.format(
      "%-8s %3d %9s  %02d",
      this.name, this.dep, value_fmt.format( this.activity ), this.group
      );
  } // toPrettyString


//----------------------------------------------------------------
//  METHOD:         toString()                                     -
/*****************************************************************
*****************************************************************/
@Override
public String
toString()
  {
  return
  String.format(
      "%s,%d,%-8g,%d",
      this.name, this.dep, this.activity, this.group
      );
  } // toString

```

```

    //------------------------------------------------------------------
    //  METHOD:         valueOf()                                        -
    /*******************************************************************
    *******************************************************************/
    public void
    valueOf( String line )
      {
      String[]  tokens = line.split( "[\\s,]+" );

      this.name = tokens.length > 0 ? tokens[ 0 ] : "";
      this.dep = tokens.length > 1 ? DataUtils.parseInt( tokens[ 1 ], 0 ) : 0;
      this.activity =
          tokens.length > 2 ? DataUtils.parseDouble( tokens[ 2 ], 0.0 ) : 0.0;
      this.group = tokens.length > 3 ?
          DataUtils.parseInt( tokens[ 3 ], -1 ) : -1;
      } // valueOf


    //------------------------------------------------------------------
    //  METHOD:         readArray()                                      -
    /*******************************************************************
    *******************************************************************/
    public static IsotopeRec[]
    readArray( BufferedReader reader )
        throws IOException
      {
      ArrayList<IsotopeRec>  rec_list = new ArrayList<>( 1024 );
      String  line;

      while (
          (line = reader.readLine()) != null &&
          (line = line.trim()).length() > 0
          )
        {
        if ( ! line.startsWith( "Isotope" ) )
          rec_list.add( new IsotopeRec( line ) );
        }

      Collections.sort( rec_list );
      return  rec_list.toArray( new IsotopeRec[ rec_list.size() ] );
      } // readArray


    //------------------------------------------------------------------
    //  METHOD:         toHeaderString()                                 -
    /*******************************************************************
    *******************************************************************/
    public static String
    toHeaderString()
      {
      return
      String.format(
          "%-8s %3s %9s %3s",
          "Isotope", "Dep", "Activity", "Grp"
          );
      } // toHeaderString
    } // IsotopeRec


  //--------------------------------------------------------------------
  //    CLASS:          ReleaseRec                                       -
  /*********************************************************************
  *********************************************************************/
  public static class
```

```
ReleaseRec
  {
  public IsotopeRec[] base;

  public IsotopeRec[] depositors;

  public IsotopeRec[] nondepositors;

  public int          index = -1;


  //-------------------------------------------------------------------
  //  METHOD:         ReleaseRec()                                      -
  /********************************************************************
  ********************************************************************/
  public
  ReleaseRec(
      int             index,
      IsotopeRec[]    base,
      IsotopeRec[]    depositors,
      IsotopeRec[]    nondepositors
      )
      throws IOException
    {
    this.index = index;
    this.base = base;
    this.depositors = depositors;
    this.nondepositors = nondepositors;
    } // ReleaseRec


  //-------------------------------------------------------------------
  //  METHOD:         equals()                                          -
  /********************************************************************
  ********************************************************************/
  @Override
  public boolean
  equals( Object that )
    {
    boolean  equal = false;

    if ( that instanceof ReleaseRec )
      {
      ReleaseRec  rthat = (ReleaseRec)that;

      equal =
          this.index == rthat.index &&
          Arrays.equals( this.base, rthat.base ) &&
          Arrays.equals( this.depositors, rthat.depositors ) &&
          Arrays.equals( this.nondepositors, rthat.nondepositors );
      } // if a ReleaseRec

    return  equal;
    } // equals


  //-------------------------------------------------------------------
  //  METHOD:         write()                                           -
  /********************************************************************
  ********************************************************************/
  public void
  write( PrintStream writer )
      throws IOException
    {
    DecimalFormat  value_fmt = new DecimalFormat( "0.0##E00" );

    writer.printf(
        "++release_base_activity pir_ndx=%d:%n%s%n",
        this.index, IsotopeRec.toHeaderString()
        );
    for ( IsotopeRec iso_rec : this.base )
      writer.println( iso_rec.toPrettyString( value_fmt ) );

    writer.printf(
        "%n+++depositors %d:%n%s%n",
        this.index, IsotopeRec.toHeaderString()
        );
    for ( IsotopeRec iso_rec : this.depositors )
      writer.println( iso_rec.toPrettyString( value_fmt ) );

    writer.printf(
        "%n+++non-depositors %d:%n%s%n",
        this.index, IsotopeRec.toHeaderString()
        );
    for ( IsotopeRec iso_rec : this.nondepositors )
      writer.println( iso_rec.toPrettyString( value_fmt ) );
    } // write


  //---------------------------------------------------------------------
  //  METHOD:         readNext()                                        -
  /*********************************************************************
  * @return           next object read or null if no more
  *********************************************************************/
  public static ReleaseRec
  readNext( BufferedReader reader )
      throws IOException
    {
    ReleaseRec  rec = null;
    int  index = -1;
    IsotopeRec[]  base_acts = null, deps = null, non_deps = null;

              // Outer Read Loop
              //
    for ( boolean finished = false; ! finished; )
      {
      String  line;
      Matcher  m;

                      // Check for Finished Sentinel
                      //
      if (
          (line = reader.readLine()) == null ||
          (line = line.trim()).contains( "+release: " ) ||
          (line = line.trim()).contains( "+activityTableInputs: " )
          )
        finished = true;

      else if (
          line.startsWith( "++" ) &&
          (m = PATTERN_releaseBaseActivity.matcher( line )).matches()
          )
        {
        index = DataUtils.parseInt( m.group( 1 ), -1 );
        base_acts = IsotopeRec.readArray( reader );
        }

      else if ( line.contains( "++depositor list" ) )
        deps = IsotopeRec.readArray( reader );

      else if ( line.contains( "++non-depositor list" ) )
        {
        non_deps = IsotopeRec.readArray( reader );
        finished = true;
        }
      } // for outer loop

    if ( index >= 0 )
      rec = new ReleaseRec( index, base_acts, deps, non_deps );

    return  rec;
    } // readNext
  } // ReleaseRec
} // NfacLog
```

C. RADFILE SOURCE LISTING

```
// $Id$
//-----------------------------------------------------------------------
//      NAME:           RadFile.java                                    -
//      HISTORY:                                                        -
//              2015-05-19      leerw@ornl.gov
//        Added materialPIRIndexIncr property, used in
//        readMaterialFactorsItem(), and incrPIRIndex().
//              2015-05-18      leerw@ornl.gov
//        Material index name is the first 10 chars, ci{nd,dp}NNssMM
//        Calling TestUtils.equals() on factors in MaterialDoseFactor.equals().
//              2014-12-12      leerw@ornl.gov
//        Fixed getMaterialIndexName() to always return the first
//        8 chars.
//              2014-10-03      leerw@ornl.gov
//        Re-structured to use SortedMap for easy comparisons.
//              2014-10-02      leerw@ornl.gov
//              2014-10-01      leerw@ornl.gov
//              2014-09-30      leerw@ornl.gov
//              2014-09-25      leerw@ornl.gov
//        Started readMaterialIsotopeActivityFactors().
//              2014-09-15      leerw@ornl.gov
//              2014-09-12      leerw@ornl.gov
//-----------------------------------------------------------------------
package mil.dtra.hpac.models.nfac.test;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.Reader;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.TreeMap;

import java.util.zip.GZIPInputStream;

import mil.dtra.hpac.util.DataUtils;


//-----------------------------------------------------------------------
//      CLASS:          RadFile                                         -
/************************************************************************
* Captures the NFAC log "FINE" level output showing inventories and
* activities.
************************************************************************/
public class
RadFile
  {
                // Constants
                //

  public final static
```

```
    String              BASE_NAME = "nfac.test.RadFile";
                                //RadFile.class.getPackage().getName();

  public final static
    double[]            EMPTY_doubles = new double[ 0 ];

  public final static
    String              HEADER_LINE =
        "#====================================================" +
        System.getProperty( "line.separator" );

  public final static
    String              PROP_debug = BASE_NAME + ".debug";

/*
  public final static
    String              REGEX_releaseBaseActivity =
"^\\Q++\\Erelease_base_activity, pir_ndx=([0-9]+)/([0-9]+), decay_time=([\\d\\.]+).*$";
// "^\\Q++\\Erelease_base_activity, pir_ndx=([0-9]+):$";

  public final static
    Pattern             PATTERN_releaseBaseActivity =
        Pattern.compile( REGEX_releaseBaseActivity );

  public final static
    Pattern             PATTERN_ws = Pattern.compile( "[\\s,]+" );
*/


                // Class Attributes
                //

/*
  protected final static
    Logger              fLogger_ =
        Logger.getLogger( SourceTermTableTest.class.getPackage().getName() );
*/


                // Object Attributes
                //

  private
    List<Tuple<String,int[]>>  fComponentDoses;
    //AtomicReference<List<Tuple<String,int[]>>>  fComponentDosesRef;

  private
    Map<String,Integer> fComponentDosesIndex;
    //AtomicReference<Map<String,Integer>>  fComponentDosesIndexRef;

  private
    String[]                    fHeaderLines;

                                /**
                                * by index name
                                */
  private
    SortedMap<String,MaterialDoseFactor>  fMaterialDoseFactors;
    //AtomicReference<List<MaterialDoseFactor>>  fMaterialDoseFactorsRef;

                                /**
                                * by index name
                                */
  private
    SortedMap<String,MaterialDoseFactor>  fMaterialIsotopeActivityFactors;
    //List<Tuple<String,int[]>>  fMaterialIsotopeActivityFactors;
    //AtomicReference<Map<String,int[]>>  fMaterialIsotopeActivityFactorsRef;

                                /**
                                * by index name of full name
                                */
  private
    SortedMap<String,String>  fMaterialNames;
    //AtomicReference<List<String>>  fMaterialNamesRef;

  private
    int                 fMaterialPIRIndexIncr;

  private
    List<Tuple<String,int[]>>  fPlottedDoses;
    //AtomicReference<List<Tuple<String,int[]>>>  fPlottedDosesRef;

  private
    Map<String,Integer> fPlottedDosesIndex;
    //AtomicReference<Map<String,Integer>>  fPlottedDosesIndexRef;

  private
    List<String>        fPlottedIsotopes;
    //AtomicReference<List<String>>  fPlottedIsotopes;

  private
    Map<String,Integer> fPlottedIsotopesIndex;
    //AtomicReference<Map<String,Integer>>  fPlottedIsotopesIndexRef;


                // Object Methods
                //


  //----------------------------------------------------------------------
  //    METHOD:         RadFile()                                         -
  /**********************************************************************
  **********************************************************************/
  protected
  RadFile( int mat_pir_index_incr )
    {
    setMaterialPIRIndexIncr( mat_pir_index_incr );

    setComponentDoses( new ArrayList<Tuple<String,int[]>>( 0 ) );
    setHeaderLines( null );
    setMaterialDoseFactors( new TreeMap<String,MaterialDoseFactor>() );
    setMaterialIsotopeActivityFactors( new TreeMap<String,MaterialDoseFactor>() );
    setMaterialNames( new TreeMap<String,String>() );
    setPlottedDoses( new ArrayList<Tuple<String,int[]>>( 0 ) );
    setPlottedIsotopes( new ArrayList<String>( 0 ) );
/*
    fComponentsDosesRef = new AtomicReference<List<Tuple<String,int[]>>>();
    fMaterialDoseFactorsRef = new AtomicReference<List<MaterialDoseFactor>>();
    fMaterialIsotopeActivityFactorsRef =
        new AtomicReference<List<Tuple<String,int[]>>>();
    fMaterialNamesRef = new AtomicReference<List<String>>();
    fPlottedDosesRef = new AtomicReference<List<Tuple<String,int[]>>>();
    fPlottedIsotopes = new AtomicReference<List<String>>();
*/
    } // RadFile


  //----------------------------------------------------------------------
  //    METHOD:         RadFile()                                         -
  /**********************************************************************
```

```
*********************************************************************/
public
RadFile( File file )
    throws IOException
  {
  this( file, 0 );
  } // RadFile


//--------------------------------------------------------------------
//    METHOD:         RadFile()                                      -
/*********************************************************************
*********************************************************************/
public
RadFile( InputStream input )
    throws IOException
  {
  this( input, 0 );
  } // RadFile


//--------------------------------------------------------------------
//    METHOD:         RadFile()                                      -
/*********************************************************************
*********************************************************************/
public
RadFile( File file, int mat_pir_index_incr )
    throws IOException
  {
  this( mat_pir_index_incr );
  read( file );
  } // RadFile


//--------------------------------------------------------------------
//    METHOD:         RadFile()                                      -
/*********************************************************************
*********************************************************************/
public
RadFile( InputStream input, int mat_pir_index_incr )
    throws IOException
  {
  this( mat_pir_index_incr );
  read( input );
  } // RadFile


//--------------------------------------------------------------------
//    METHOD:         equals()                                       -
/*********************************************************************
*********************************************************************/
@Override
public boolean
equals( Object that )
  {
  return  !(that instanceof RadFile) ? false : equals( (RadFile)that );
  } // equals


//--------------------------------------------------------------------
//    METHOD:         equals()                                       -
/*********************************************************************
*********************************************************************/
public boolean
equals( RadFile that )
```

```
{
boolean  equal = false;

ArrayList<Tuple<String,int[]>>  this_comp_doses =
    this.fComponentDoses == null ? null :
    new ArrayList<>( this.fComponentDoses );
Collections.sort( this_comp_doses );
ArrayList<Tuple<String,int[]>>  that_comp_doses =
    that.fComponentDoses == null ? null :
    new ArrayList<>( that.fComponentDoses );
Collections.sort( that_comp_doses );

ArrayList<Tuple<String,int[]>>  this_plotted_doses =
    this.fPlottedDoses == null ? null :
    new ArrayList<>( this.fPlottedDoses );
Collections.sort( this_plotted_doses );
ArrayList<Tuple<String,int[]>>  that_plotted_doses =
    that.fPlottedDoses == null ? null :
    new ArrayList<>( that.fPlottedDoses );
Collections.sort( that_plotted_doses );

ArrayList<String>  this_plotted_isos =
    this.fPlottedIsotopes == null ? null :
    new ArrayList<>( this.fPlottedIsotopes );
Collections.sort( this_plotted_isos );
ArrayList<String>  that_plotted_isos =
    that.fPlottedIsotopes == null ? null :
    new ArrayList<>( that.fPlottedIsotopes );
Collections.sort( that_plotted_isos );

Set<String>  this_mat_names =
    this.fMaterialNames == null ? null : this.fMaterialNames.keySet();
Set<String>  that_mat_names =
    that.fMaterialNames == null ? null : that.fMaterialNames.keySet();

if ( Boolean.getBoolean( PROP_debug ) )
  {
  boolean[]  eq = new boolean[ 7 ];

  eq[ 0 ] = TestUtils.deepEquals( this.fHeaderLines, that.fHeaderLines );
  System.err.printf( "[RadFile.equals] headerLines=%b%n", eq[ 0 ] );

  eq[ 1 ] = TestUtils.deepEquals( this_comp_doses, that_comp_doses );
  System.err.printf( "[RadFile.equals] componentDoses=%b%n", eq[ 1 ] );

  eq[ 2 ] = TestUtils.deepEquals( this_plotted_doses, that_plotted_doses );
  System.err.printf( "[RadFile.equals] plottedDoses=%b%n", eq[ 2 ] );

  eq[ 3 ] = TestUtils.deepEquals( this_plotted_isos, that_plotted_isos );
  System.err.printf( "[RadFile.equals] plottedIsotopes=%b%n", eq[ 3 ] );

  eq[ 4 ] = TestUtils.deepEquals( this_mat_names, that_mat_names );
  System.err.printf( "[RadFile.equals] materialNames=%b%n", eq[ 4 ] );

  eq[ 5 ] = TestUtils.
      deepEquals( this.fMaterialDoseFactors, that.fMaterialDoseFactors );
  System.err.printf( "[RadFile.equals] materialDoseFactors=%b%n", eq[ 5 ] );

  eq[ 6 ] = TestUtils.deepEquals(
      this.fMaterialIsotopeActivityFactors,
      that.fMaterialIsotopeActivityFactors
      );
  System.err.printf(
      "[RadFile.equals] materialIsotopeActivityFactors=%b%n",
      eq[ 6 ]
```

```
          );

      equal = true;
      for ( int i = 0; i < eq.length && equal; i++ )
        equal &= eq[ i ];
      }

    else
      {
      equal =
          TestUtils.deepEquals( this.fHeaderLines, that.fHeaderLines ) &&
          TestUtils.deepEquals( this_comp_doses, that_comp_doses ) &&
          TestUtils.deepEquals( this_plotted_doses, that_plotted_doses ) &&
          TestUtils.deepEquals( this_plotted_isos, that_plotted_isos ) &&
          TestUtils.deepEquals( this_mat_names, that_mat_names ) &&
          TestUtils.deepEquals(
              this.fMaterialDoseFactors,
              that.fMaterialDoseFactors
              ) &&
          TestUtils.deepEquals(
              this.fMaterialIsotopeActivityFactors,
              that.fMaterialIsotopeActivityFactors
              );
      }

    return  equal;
    } // equals


  //------------------------------------------------------------------
  //    METHOD:         getComponentDoses()                           -
  /******************************************************************
  * @return             object reference
  ******************************************************************/
  public List<Tuple<String,int[]>>
  getComponentDoses()
    {
    return  fComponentDoses;
    //return  getComponentDosesRef().get();
    } // getComponentDoses


  //------------------------------------------------------------------
  //    METHOD:         getComponentDosesIndex()                      -
  /******************************************************************
  * Map by name of 1-based indices.
  * @return             object reference
  ******************************************************************/
  public Map<String,Integer>
  getComponentDosesIndex()
    {
    return  fComponentDosesIndex;
    } // getComponentDosesIndex


  //------------------------------------------------------------------
  //    METHOD:         getComponentDosesRef()                        -
  /******************************************************************
  * @return             object reference
  ******************************************************************/
/*
  protected AtomicReference<List<Tuple<String,int[]>>>
  getComponentDosesRef()
    {
    return  fComponentDosesRef;
    } // getComponentDosesRef
*/


  //------------------------------------------------------------------
  //    METHOD:         getHeaderLines()                             -
  /*******************************************************************
  * @return             array reference, possibly null
  *******************************************************************/
  public String[]
  getHeaderLines()
    {
    return  fHeaderLines;
    } // getHeaderLines


  //------------------------------------------------------------------
  //    METHOD:         getMaterialDoseFactors()                     -
  /*******************************************************************
  * Map by indexname of records.
  * @return             object reference
  *******************************************************************/
  public SortedMap<String,MaterialDoseFactor>
  getMaterialDoseFactors()
    {
    return  fMaterialDoseFactors;
    //return  getMaterialDoseFactorsRef().get();
    } // getMaterialDoseFactors


  //------------------------------------------------------------------
  //    METHOD:         getMaterialIsotopeActivityFactors()          -
  /*******************************************************************
  * Map by indexname of records.
  * @return             object reference
  *******************************************************************/
  public SortedMap<String,MaterialDoseFactor>
  getMaterialIsotopeActivityFactors()
    {
    return  fMaterialIsotopeActivityFactors;
    //return  getMaterialIsotopeActivityFactorsRef().get();
    } // getMaterialIsotopeActivityFactors


  //------------------------------------------------------------------
  //    METHOD:         getMaterialNames()                           -
  /*******************************************************************
  * Hash by index material name of full material names.
  * @return             object reference
  *******************************************************************/
  public Map<String,String>
  getMaterialNames()
    {
    return  fMaterialNames;
    //return  getMaterialNamesRef().get();
    } // getMaterialNames


  //------------------------------------------------------------------
  //    METHOD:         getMaterialPIRIndexIncr()                    -
  /*******************************************************************
  *******************************************************************/
  public int
  getMaterialPIRIndexIncr()
    {
```

```
  return  fMaterialPIRIndexIncr;
  } // getMaterialPIRIndexIncr


//------------------------------------------------------------------
//    METHOD:         getPlottedDoses()                            -
/*******************************************************************
* @return             object reference
*******************************************************************/
public List<Tuple<String,int[]>>
getPlottedDoses()
  {
  return  fPlottedDoses;
  //return  getPlottedDosesRef().get();
  } // getPlottedDoses


//------------------------------------------------------------------
//    METHOD:         getPlottedDosesIndex()                       -
/*******************************************************************
* Map by name of 1-based indices.
* @return             object reference
*******************************************************************/
public Map<String,Integer>
getPlottedDosesIndex()
  {
  return  fPlottedDosesIndex;
  } // getPlottedDosesIndex


//------------------------------------------------------------------
//    METHOD:         getPlottedIsotopes()                         -
/*******************************************************************
* @return             object reference
*******************************************************************/
public List<String>
getPlottedIsotopes()
  {
  return  fPlottedIsotopes;
  //return  getPlottedIsotopesRef().get();
  } // getPlottedIsotopes


//------------------------------------------------------------------
//    METHOD:         getPlottedIsotopesIndex()                    -
/*******************************************************************
* Map by name of 1-based indices.
* @return             object reference
*******************************************************************/
public Map<String,Integer>
getPlottedIsotopesIndex()
  {
  return  fPlottedIsotopesIndex;
  } // getPlottedIsotopesIndex


//------------------------------------------------------------------
//    METHOD:         read()                                       -
/*******************************************************************
* Calls {@link #read( InputStream )}.
*******************************************************************/
public void
read( File file )
    throws IOException
  {
```

```
  //FileInputStream  file_in = new FileInputStream( file );
  InputStream  input = new FileInputStream( file );

  if ( file.getName().endsWith( ".gz" ) )
    input = new GZIPInputStream( input );

  try
    { read( input ); }
  finally
    { input.close(); }
  } // read


//----------------------------------------------------------------------
//    METHOD:        read()                                           -
/***********************************************************************
***********************************************************************/
public void
read( InputStream input )
    throws IOException
  {
  read( new InputStreamReader( input ) );
  } // read


//----------------------------------------------------------------------
//    METHOD:        read()                                           -
/***********************************************************************
* Implementation method.
***********************************************************************/
public void
read( Reader reader_in )
    throws IOException
  {
  BufferedReader  reader =
     (reader_in instanceof BufferedReader) ?
     (BufferedReader)reader_in : new BufferedReader( reader_in );

  setHeaderLines( new String[]{ reader.readLine(), reader.readLine() } );

  ArrayList<Tuple<String,int[]>>  component_doses = new ArrayList<>( 128 );
  readComponentDoses( reader, component_doses );
  setComponentDoses( component_doses );

  ArrayList<String>  component_dose_names =
      new ArrayList<>( component_doses.size() );
  for ( Tuple<String,int[]> item : component_doses )
    component_dose_names.add( item.x );

  ArrayList<Tuple<String,int[]>>  plotted_doses = new ArrayList<>( 128 );
  readPlottedDoses( reader, plotted_doses );
  setPlottedDoses( plotted_doses );

  TreeMap<String,MaterialDoseFactor>  mat_dose_factors = new TreeMap<>();
  TreeMap<String,String>  mat_names = new TreeMap<>();
  readMaterialDoseFactors(
      reader, component_dose_names, mat_dose_factors, mat_names
      );
  setMaterialDoseFactors( mat_dose_factors );
  setMaterialNames( mat_names );

  ArrayList<String>  plotted_isotopes = new ArrayList<>( 128 );
  readPlottedIsotopes( reader, plotted_isotopes );
  setPlottedIsotopes( plotted_isotopes );

  TreeMap<String,MaterialDoseFactor>  mat_iso_activity_factors =
      new TreeMap<>();
  readMaterialIsotopeActivityFactors(
      reader, getPlottedIsotopes(), mat_iso_activity_factors
      );
  setMaterialIsotopeActivityFactors( mat_iso_activity_factors );
  } // read


//----------------------------------------------------------------------
//    METHOD:         readComponentDoses()                            -
/***********************************************************************
***********************************************************************/
protected void
readComponentDoses( BufferedReader reader, List<Tuple<String,int[]>> list )
    throws IOException
  {
  String  line, line2;

  while ( !(
      (line = reader.readLine()) == null || line.contains( "Component Doses" )
      ) )
    ;

  reader.readLine();

  if ( (line = reader.readLine()) != null )
    {
    int  count = DataUtils.parseInt( line.trim() );

    for ( int i = 0; i < count; i++ )
      {
      if (
          (line = reader.readLine()) != null &&
          (line2 = reader.readLine()) != null
          )
        {
        String  name = line.trim();
        int  value = DataUtils.parseInt( line2.trim() );
        list.add( new Tuple<String,int[]>( name, new int[]{ value } ) );
        } // if both lines
      } // for each dose
    } // if we have a count line
  } // readComponentDoses


//----------------------------------------------------------------------
//    METHOD:         readMaterialDoseFactors()                       -
/***********************************************************************
***********************************************************************/
protected void
readMaterialDoseFactors(
    BufferedReader    reader,
    List<String>      ref_names,
    SortedMap<String,MaterialDoseFactor>  factors_map,
    Map<String,String>  names
    )
    throws IOException
  {
  String  line, line2;

  while ( !(
      (line = reader.readLine()) == null ||
      line.contains( "Material Dose Factors" )
      ) )
    ;

  reader.readLine();

  if ( (line = reader.readLine()) != null )
    {
    int  count = DataUtils.parseInt( line.trim() );

    for ( int i = 0; i < count; i++ )
      readMaterialFactorsItem( reader, ref_names, factors_map, names, 3 );
    } // if count line
  } // readMaterialDoseFactors


//------------------------------------------------------------------
//    METHOD:         readMaterialFactorsItem()                       -
/******************************************************************
******************************************************************/
protected void
readMaterialFactorsItem(
    BufferedReader    reader,
    List<String>      ref_names,
    SortedMap<String,MaterialDoseFactor>  factors_map,
    Map<String,String>  names,
    int               numbers_count
    )
    throws IOException
  {
  String  line;

  if ( (line = reader.readLine()) != null )
    {
    String  full_name = line.trim();
    String  index_name = getMaterialIndexName( full_name );

    if ( getMaterialPIRIndexIncr() != 0 )
      {
      full_name = incrPIRIndex( full_name, getMaterialPIRIndexIncr() );
      index_name = incrPIRIndex( index_name, getMaterialPIRIndexIncr() );
      }

                      // Update Names Map
                      //
    if ( names != null )
      names.put( index_name, full_name );

                      // Times
                      //
    double[]  times = EMPTY_doubles;
    int  times_count = 0;

    if ( (line = reader.readLine()) != null )
      {
      times_count = DataUtils.parseInt( line.trim() );
      times = readValuesBlock( reader, times_count );
      }

                      // Mystery Numbers
                      //
    double[]  numbers = null;
    if ( numbers_count > 0 )
      {
      numbers = new double[ numbers_count ];
      for ( int k = 0; k < numbers.length; k++ )
        {
```

```
          numbers[ k ] =
              (line = reader.readLine()) == null ? 0.0 :
              DataUtils.parseDouble( line.trim(), 0.0 );
          }
        }

                        // Factors
                        //
      //double[][]  factors = null;
      //int[]  factor_indexes = null;
      TreeMap<String,double[]>  factors = null;

      if ( (line = reader.readLine()) != null )
        {
        int  factors_count = DataUtils.parseInt( line.trim() );
        //factors = new double[ factors_count ][];
        //factor_indexes = new int[ factors_count ];
        //Arrays.fill( factors, EMPTY_doubles );
        factors = new TreeMap<String,double[]>();

        for ( int i = 0; i < factors_count; i++ )
          {
          int  mat_index;

          if (
              (line = reader.readLine()) != null &&
              (mat_index = DataUtils.parseInt( line.trim(), 0 )) > 0 &&
              mat_index <= ref_names.size()
              )
            {
            factors.put(
                ref_names.get( mat_index - 1 ),
                readValuesBlock( reader, times_count )
                );
/*
            factors.put(
                new Integer( mat_index ),
                readValuesBlock( reader, times_count )
                );
*/
            }
          }
        } // if factors count line

                        // Update List
                        //
      MaterialDoseFactor  rec = new MaterialDoseFactor(
          index_name, full_name,
          times, factors, numbers
          //times, factor_indexes, factors, numbers
          );
      factors_map.put( index_name, rec );
      } // if name line read
    } // readMaterialFactorsItem


  //-----------------------------------------------------------------------
  //    METHOD:         readMaterialIsotopeActivityFactors()            -
  /***********************************************************************
  ***********************************************************************/
  protected void
  readMaterialIsotopeActivityFactors(
      BufferedReader    reader,
      List<String>      ref_names,
      SortedMap<String,MaterialDoseFactor>  factors_map
      )
      throws IOException
    {
    String  line, line2;

    while ( !(
        (line = reader.readLine()) == null ||
        line.contains( "Material Isotope Activity Factors" )
        ) )
      ;

    reader.readLine();

    if ( (line = reader.readLine()) != null )
      {
      int  count = DataUtils.parseInt( line.trim() );

      for ( int i = 0; i < count; i++ )
        readMaterialFactorsItem( reader, ref_names, factors_map, null, 0 );
      } // if count line
    } // readMaterialIsotopeActivityFactors


  //----------------------------------------------------------------------
  //    METHOD:         readPlottedDoses()                               -
  /***********************************************************************
  ***********************************************************************/
  protected void
  readPlottedDoses( BufferedReader reader, List<Tuple<String,int[]>> list )
      throws IOException
    {
    String  line, line2, line3;

    while ( !(
        (line = reader.readLine()) == null || line.contains( "Plotted Doses" )
        ) )
      ;

    reader.readLine();

    if ( (line = reader.readLine()) != null )
      {
      int  count = DataUtils.parseInt( line.trim() );

      for ( int i = 0; i < count; i++ )
        {
        if (
            (line = reader.readLine()) != null &&
            (line2 = reader.readLine()) != null &&
            (line3 = reader.readLine()) != null
            )
          {
          String  name = line.trim();

          String[]  tokens = line3.trim().split( "[\\s,]+" );
          int[]  value_arr = new int[ tokens.length + 1 ];
          value_arr[ 0 ] = DataUtils.parseInt( line2.trim() );

          for ( int k = 0; k < tokens.length; k++ )
            value_arr[ k + 1 ] = DataUtils.parseInt( tokens[ k ] );

          list.add( new Tuple<String,int[]>( name, value_arr ) );
          } // if both lines
        } // for each dose
      } // if we have a count line
    } // readPlottedDoses


  //-----------------------------------------------------------------------
  //    METHOD:         readPlottedIsotopes()                            -
  /***********************************************************************
  ***********************************************************************/
  protected void
  readPlottedIsotopes( BufferedReader reader, List<String> list )
      throws IOException
    {
    String  line;

    while ( !(
        (line = reader.readLine()) == null || line.contains( "Plotted Isotopes" )
        ) )
      ;

    reader.readLine();

    if ( (line = reader.readLine()) != null )
      {
      int  count = DataUtils.parseInt( line.trim() );

      for ( int i = 0; i < count; i++ )
        {
        if ( (line = reader.readLine()) != null )
          list.add( line.trim() );
        } // for each dose
      } // if we have a count line
    } // readPlottedIsotopes


  //-----------------------------------------------------------------------
  //    METHOD:         setComponentDoses()                              -
  /***********************************************************************
  * @param  value       object reference to store
  ***********************************************************************/
  public void
  setComponentDoses( List<Tuple<String,int[]>> value )
    {
    setComponentDosesRef( value );

    Map<String,Integer>  index = new HashMap<>( value.size() << 1 );

    for ( int i = 0; i < value.size(); i++ )
      index.put( value.get( i ).x, new Integer( i + 1 ) );

    setComponentDosesIndex( index );
    } // setComponentDoses


  //-----------------------------------------------------------------------
  //    METHOD:         setComponentDosesRef()                           -
  /***********************************************************************
  * @param  value       object reference to store
  ***********************************************************************/
  protected void
  setComponentDosesRef( List<Tuple<String,int[]>> value )
    {
    fComponentDoses = Collections.unmodifiableList( value );
    } // setComponentDosesRef


  //-----------------------------------------------------------------------
```

```
 //    METHOD:         setComponentDosesIndex()                       -
 /*******************************************************************
 * Map by name of 1-based indices.
 * @param  value       object reference to store
 *******************************************************************/
 protected void
 setComponentDosesIndex( Map<String,Integer> value )
   {
   fComponentDosesIndex = Collections.unmodifiableMap( value );
   } // setComponentDosesIndex


 //-------------------------------------------------------------------
 //    METHOD:         setHeaderLines()                               -
 /*******************************************************************
 * @param  value       array reference to store
 *******************************************************************/
 public void
 setHeaderLines( String[] value )
   {
   fHeaderLines = value;
   } // setHeaderLines


 //-------------------------------------------------------------------
 //    METHOD:         setMaterialDoseFactors()                       -
 /*******************************************************************
 * Records by 1-based index - 1.
 * @param  value       object reference to store
 *******************************************************************/
 public void
 setMaterialDoseFactors( SortedMap<String,MaterialDoseFactor> value )
   {
   fMaterialDoseFactors = Collections.unmodifiableSortedMap( value );
   //getMaterialDoseFactorsRef().set( Collections.unmodifiableList( value ) );
   } // setMaterialDoseFactors


 //-------------------------------------------------------------------
 //    METHOD:         setMaterialIsotopeActivityFactors()            -
 /*******************************************************************
 * Rec by plotted isotope 1-based index - 1.
 * @param  value       object reference to store
 *******************************************************************/
 public void
 setMaterialIsotopeActivityFactors( SortedMap<String,MaterialDoseFactor> value )
   {
   fMaterialIsotopeActivityFactors = Collections.unmodifiableSortedMap( value );
/*
   getMaterialIsotopeActivityFactorsRef().
       set( Collections.unmodifiableMap( value ) );
*/
   } // setMaterialIsotopeActivityFactors


 //-------------------------------------------------------------------
 //    METHOD:         setMaterialNames()                             -
 /*******************************************************************
 * Hash by index material name of full material names.
 * @param  value       object reference to store
 *******************************************************************/
 public void
 setMaterialNames( SortedMap<String,String> value )
   {
   fMaterialNames = Collections.unmodifiableSortedMap( value );
```

```
  //getMaterialNamesRef().set( Collections.unmodifiableList( value ) );
  } // setMaterialNames


//------------------------------------------------------------------------
//    METHOD:         setMaterialPIRIndexIncr()                          -
/************************************************************************
************************************************************************/
public void
setMaterialPIRIndexIncr( int value )
  {
  fMaterialPIRIndexIncr = value;
  } // setMaterialPIRIndexIncr


//------------------------------------------------------------------------
//    METHOD:         setPlottedDoses()                                  -
/************************************************************************
* @param  value       object reference to store
************************************************************************/
public void
setPlottedDoses( List<Tuple<String,int[]>> value )
  {
  //getPlottedDosesRef().set( Collections.unmodifiableMap( value ) );
  setPlottedDosesRef( value );

  Map<String,Integer>  index = new HashMap<>( value.size() << 1 );

  for ( int i = 0; i < value.size(); i++ )
    index.put( value.get( i ).x, new Integer( i + 1 ) );

  setPlottedDosesIndex( index );
  } // setPlottedDoses


//------------------------------------------------------------------------
//    METHOD:         setPlottedDosesRef()                               -
/************************************************************************
* @param  value       object reference to store
************************************************************************/
protected void
setPlottedDosesRef( List<Tuple<String,int[]>> value )
  {
  fPlottedDoses = Collections.unmodifiableList( value );
  } // setPlottedDosesRef


//------------------------------------------------------------------------
//    METHOD:         setPlottedDosesIndex()                             -
/************************************************************************
* Map by name of 1-based indices.
* @param  value       object reference to store
************************************************************************/
protected void
setPlottedDosesIndex( Map<String,Integer> value )
  {
  fPlottedDosesIndex = Collections.unmodifiableMap( value );
  } // setPlottedDosesIndex


//------------------------------------------------------------------------
//    METHOD:         setPlottedIsotopes()                               -
/************************************************************************
* @param  value       object reference to store
************************************************************************/
```

```
public void
setPlottedIsotopes( List<String> value )
  {
  setPlottedIsotopesRef( value );

  Map<String,Integer>  index = new HashMap<>( value.size() << 1 );

  for ( int i = 0; i < value.size(); i++ )
    index.put( value.get( i ), new Integer( i + 1 ) );

  setPlottedIsotopesIndex( index );
  } // setPlottedIsotopes


//------------------------------------------------------------------------
//    METHOD:         setPlottedIsotopesRef()                            -
/*************************************************************************
* @param  value       object reference to store
*************************************************************************/
protected void
setPlottedIsotopesRef( List<String> value )
  {
  fPlottedIsotopes = Collections.unmodifiableList( value );
  } // setPlottedIsotopesRef


//------------------------------------------------------------------------
//    METHOD:         setPlottedIsotopesIndex()                          -
/*************************************************************************
* Map by name of 1-based indices.
* @param  value       object reference to store
*************************************************************************/
protected void
setPlottedIsotopesIndex( Map<String,Integer> value )
  {
  fPlottedIsotopesIndex = Collections.unmodifiableMap( value );
  } // setPlottedIsotopesIndex


//------------------------------------------------------------------------
//    METHOD:         write()                                            -
/*************************************************************************
*************************************************************************/
public void
write( PrintStream print )
    throws IOException
  {
  if ( getHeaderLines() != null )
    {
    for ( String line : getHeaderLines() )
      print.println( line );
    }

  print.printf( HEADER_LINE + "Component Doses%n" + HEADER_LINE );
  List<Tuple<String,int[]>>  comp_doses = getComponentDoses();
  writeTuples( print, comp_doses, false );

  print.printf( HEADER_LINE + "Plotted Doses%n" + HEADER_LINE );
  List<Tuple<String,int[]>>  plotted_doses = getPlottedDoses();
  writeTuples( print, plotted_doses, true );

  print.printf( HEADER_LINE + "Material Dose Factors%n" + HEADER_LINE );
  SortedMap<String,MaterialDoseFactor>  mat_dose_factors =
      getMaterialDoseFactors();
  print.printf( " %d%n", mat_dose_factors.size() );
  for ( MaterialDoseFactor mat_dose_factor : mat_dose_factors.values() )
    mat_dose_factor.write( print, getComponentDosesIndex() );

  print.printf( HEADER_LINE + "Plotted Isotopes%n" + HEADER_LINE );
  List<String>  plotted_isos = getPlottedIsotopes();
  print.printf( " %d%n", plotted_isos.size() );
  for ( String iso : plotted_isos )
    print.println( iso );

  print.printf( HEADER_LINE + "Material Isotope Activity Factors%n" + HEADER_LINE );
  SortedMap<String,MaterialDoseFactor>  mat_iso_factors = getMaterialIsotopeActivityFactors();
  print.printf( " %d%n", mat_iso_factors.size() );
  for ( MaterialDoseFactor mat_iso_factor : mat_iso_factors.values() )
    mat_iso_factor.write( print, getPlottedIsotopesIndex() );
  } // write


//----------------------------------------------------------------------
//    METHOD:         writeTuples()                                   -
/***********************************************************************
***********************************************************************/
public void
writeTuples(
    PrintStream       print,
    List<Tuple<String,int[]>>  list,
    boolean           multi_line
    )
    throws IOException
  {
  if ( list != null )
    {
    print.printf( " %d%n", list.size() );
    for ( Tuple<String,int[]> tuple : list )
      {
      int  st_ndx = 0;

      print.println( tuple.x );
      if ( tuple.y.length > 1 && multi_line )
        {
        print.printf( " %d%n", tuple.y[ 0 ] );
        st_ndx = 1;
        }

      while ( st_ndx < tuple.y.length )
        print.printf( " %d", tuple.y[ st_ndx++ ] );
      print.println();
      } // for
    } // if
  } // writeTuples


              // Class Methods
              //


//----------------------------------------------------------------------
//    METHOD:         getMaterialIndexName()                          -
/***********************************************************************
***********************************************************************/
public static String
getMaterialIndexName( String full_name )
  {
  return
      full_name == null ? null :
      full_name.length() >= 10 ? full_name.substring( 0, 10 ) :
```

```
      full_name;
  } // getMaterialIndexName


//------------------------------------------------------------------
//    METHOD:         incrPIRIndex()                               -
/*******************************************************************
*******************************************************************/
public static String
incrPIRIndex( String mat_name, int incr )
  {
  String  result = mat_name;

  if ( mat_name.length() >= 6 )
    {
    int  cur_index = DataUtils.parseInt( mat_name.substring( 4, 6 ), 0 );
    result = String.format(
        "%s%02d%s",
        mat_name.substring( 0, 4 ),
        Math.max( 0, Math.min( 99, cur_index + incr ) ),
        mat_name.substring( 6 )
        );
    } // if minimum required length

  return  result;
  } // incrPIRIndex


//------------------------------------------------------------------
//    METHOD:         main()                                       -
/*******************************************************************
*******************************************************************/
public static void
main( String[] argv )
  {
  try
    {
    File  one = null, two = null;
    int  pir_incr = 0;
    boolean  usage_flag = false;

    for ( int i = 0; i < argv.length; i++ )
      {
      if ( "-h".equals( argv[ i ] ) )
        usage_flag = true;

      else if ( "-i".equals( argv[ i ] ) && i < argv.length -1 )
        pir_incr = DataUtils.parseInt( argv[ ++i ], 0 );

      else if ( one == null )
        one = new File( argv[ i ] );
        //one = new RadFile( new File( argv[ i ] ) );

      else
        two = new File( argv[ i ] );
        //two = new RadFile( new File( argv[ i ] ) );
      } // for

    if ( usage_flag || (one == null && two == null) )
      System.out.println( "Usage: RadFile [ -i pir_incr ] file1 [ file2 ]" );

    else if ( two == null )
      new RadFile( one, pir_incr ).write( System.out );

    else
```

```
      {
      RadFile
          rone = new RadFile( one, pir_incr ),
          rtwo = new RadFile( two );
      System.out.println( rone.equals( rtwo ) ? "EQUAL" : "not equal" );
      }
    } // try

  catch ( Exception ex )
    {
    ex.printStackTrace( System.err );
    }
  } // main


//----------------------------------------------------------------------
//    METHOD:         toDecimal()                                     -
/***********************************************************************
***********************************************************************/
public static String
toDecimal( int width, int dec_index, double value )
  {
  StringBuilder  buffer = new StringBuilder( width + 16 );
  String  value_str = Double.toString( value );
  int  dot_index = value_str.indexOf( "." );

  String  left, right;
  if ( dot_index < 0 )
    {
    left = value_str;
    right = "";
    }
  else
    {
    left = value_str.substring( 0, dot_index );
    right = value_str.substring( dot_index );
    }

  int  padding = dec_index - left.length() - 1;
  if ( padding > 0 )
    buffer.append( String.format( String.format( "%%%ds", padding ), " " ) );

  buffer.append( left ).append( right );

  padding = width - buffer.length();
  if ( padding > 0 )
    buffer.append( String.format( String.format( "%%%ds", padding ), " " ) );

  value_str = buffer.toString();
  if ( value_str.length() > width )
    {
    if ( value_str.contains( "E" ) || value_str.contains( "e" ) )
      {
      int  count = 0, len = value_str.length();
      while ( value_str.charAt( count ) == ’ ’ && len > width )
        {
        count++;
        len--;
        }
      value_str = value_str.substring( count );
      }
    else
      value_str = value_str.substring( 0, width );
    } // if must shift

  return  value_str;
  } // toDecimal


//------------------------------------------------------------------
//    METHOD:         readValuesBlock()                            -
/*******************************************************************
*******************************************************************/
public static double[]
readValuesBlock( BufferedReader reader, int count )
    throws IOException
  {
  double[]  values = new double[ count ];
  String  line;

  for ( int ndx = 0; ndx < count && (line = reader.readLine()) != null; )
    {
    for ( String token : line.trim().split( "[\\s,]+" ) )
      {
      values[ ndx++ ] = DataUtils.parseDouble( token );
      if ( ndx >= count )  break;
      }
    } // for lines

  return  values;
  } // readValuesBlock


//------------------------------------------------------------------
//    METHOD:         writeValuesBlock()                           -
/*******************************************************************
*******************************************************************/
public static void
writeValuesBlock( PrintStream print, double[] values )
    throws IOException
  {
  if ( values != null && values.length > 0 )
    {
    int  count = 0;
    for ( double value : values )
      {
      print.printf( "%20s", toDecimal( 20, 10, value ) );
      if ( ++count >= 6 )
        {
        print.println();
        count = 0;
        }
      }

    if ( count != 0 )
      print.println();
    } // if values
  } // writeValuesBlock


              // Inner Classes
              //


//------------------------------------------------------------------
//    CLASS:          MaterialDoseFactor                           -
/*******************************************************************
*******************************************************************/
public static class
MaterialDoseFactor
```

```
  implements Comparable<MaterialDoseFactor>
{
//public final SortedMap<Integer,double[]>  factors;
public final SortedMap<String,double[]>  factors;

public final String  fullname;

public final String  indexname;

public final double[]  numbers;

public final double[]  times;


//------------------------------------------------------------------
//  METHOD:         MaterialDoseFactor.MaterialDoseFactor()         -
/*******************************************************************
*******************************************************************/
public
MaterialDoseFactor(
    String          index_name,
    String          full_name,
    double[]        times,
    //SortedMap<Integer,double[]>  factors,
    SortedMap<String,double[]>  factors,
    double[]        numbers
    )
  {
  this.factors = factors;
  this.fullname = full_name;
  this.indexname = index_name;
  this.numbers = numbers;
  this.times = times;
  } // MaterialDoseFactor


//------------------------------------------------------------------
//  METHOD:         compareTo()                                     -
/*******************************************************************
* Compares based on indexname.
*******************************************************************/
public int
compareTo( MaterialDoseFactor that )
  {
  return
      this.indexname == null ? (that.indexname == null ? 0 : -1) :
      that.indexname == null ? 1 :
      this.indexname.compareTo( that.indexname );
  } // compareTo


//------------------------------------------------------------------
//  METHOD:         MaterialDoseFactor.equals()                     -
/*******************************************************************
*******************************************************************/
public boolean
equals( Object that )
  {
  boolean  equal = false;

  if ( that instanceof MaterialDoseFactor )
    {
    MaterialDoseFactor  mf = (MaterialDoseFactor)that;

    equal =
        //x no DataUtils.equals( this.fullname, mf.fullname ) &&
        DataUtils.equals( this.indexname, mf.indexname ) &&
        Arrays.equals( this.times, mf.times ) &&
        Arrays.equals( this.numbers, mf.numbers ) &&
        TestUtils.getInstance().equals( this.factors, mf.factors );
        //TestUtils.deepEquals( this.factors, mf.factors );
    } // if

  return  equal;
  } // equals


//----------------------------------------------------------------------
//  METHOD:         MaterialDoseFactor.write()                         -
/**********************************************************************
***********************************************************************/
public void
write( PrintStream print, Map<String,Integer> ref_index_map )
    throws IOException
  {
  print.println( this.fullname );

  if ( this.times == null )
    print.println( " 0" );
  else
    {
    print.printf( " %d%n", this.times.length );
    writeValuesBlock( print, this.times );
    } // if we have times

  if ( this.numbers == null )
    ;

  else if ( this.numbers.length >= 3 )
    {
    print.printf(
        " %d%n%s%n %d%n",
        (int)this.numbers[ 0 ],
        toDecimal( 20, 10, this.numbers[ 1 ] ),
        (int)this.numbers[ 2 ]
        );
    }

  else
    {
    for ( double cur_number : numbers )
      print.printf( " %d%n", (int)cur_number );
    }

  if ( this.factors == null || this.factors.size() == 0 )
    print.println( " 0" );
  else
    {
    print.printf( " %d%n", this.factors.size() );
    for ( Map.Entry<String,double[]> item : this.factors.entrySet() )
      {
      if ( item.getValue() != null )
        {
        Integer  ref_index = ref_index_map.get( item.getKey() );

        print.printf(
            " %d%n",
            ref_index == null ? 0 : ref_index.intValue()
            );
        writeValuesBlock( print, item.getValue() );
        }
      }
/*
    for ( Map.Entry<Integer,double[]> item : this.factors.entrySet() )
      {
      if ( item.getValue() != null )
        {
        print.printf( " %d%n", item.getKey() );
        writeValuesBlock( print, item.getValue() );
        }
      }
*/
    } // else factors exists
  } // MaterialDoseFactor
} // MaterialDoseFactor


//---------------------------------------------------------------------
//    CLASS:          Tuple                                           -
/*********************************************************************
**********************************************************************/
public static class
Tuple<X extends Comparable<X>, Y>
  implements Comparable<Tuple<X, Y>>
{
public final X      x;
public final Y      y;


//-------------------------------------------------------------------
//  METHOD:         Tuple.Tuple()                                   -
/*******************************************************************
********************************************************************/
public
Tuple( X x, Y y )
  {
  this.x = x;
  this.y = y;
  } // Tuple


//-------------------------------------------------------------------
//  METHOD:         compareTo()                                     -
/*******************************************************************
********************************************************************/
public int
compareTo( Tuple<X, Y> that )
  {
  return
      this.x == null ? (that.x == null ? 0 : -1) :
      that.x == null ? 1 :
      this.x.compareTo( that.x );
  } // compareTo


//-------------------------------------------------------------------
//  METHOD:         Tuple.equals()                                  -
/*******************************************************************
********************************************************************/
public boolean
equals( Object that )
  {
  boolean  equal = false;

  if ( that instanceof Tuple )
```

```
        {
        Tuple  t = (Tuple)that;

        equal =
            TestUtils.deepEquals( this.x, t.x ) &&
            TestUtils.deepEquals( this.y, t.y );
        }

      return  equal;
      } // equals
    } // Tuple
  } // RadFile
```

ORNL/TM-2015/726

INTERNAL DISTRIBUTION

1. R. W. Lee
2. R. H. Morris
3. C. D. Sulfredge
4. ORNL Office of Technical Information and Classification

EXTERNAL DISTRIBUTION

5. Todd Hann, Chief, Hazards & Effects Modeling Division, Information Sciences & Applications Department, Defense Threat Reduction Agency, 8725 John J. Kingman Rd., Stop 6201, Ft Belvoir, VA 22060-6201