TRACE/SNAP User Workshop
September 30 - October 3, 2014

Hilton Garden Inn
Idaho Falls, ID

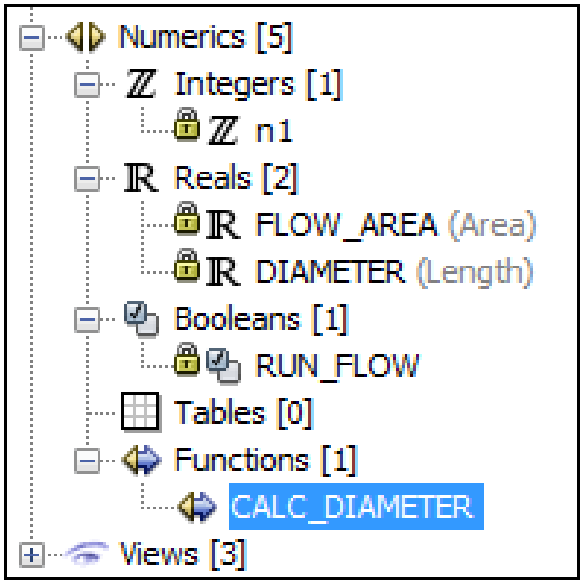
SNAP Job Streams and Post-Processing

Applied Programming Technology, Inc.

APT

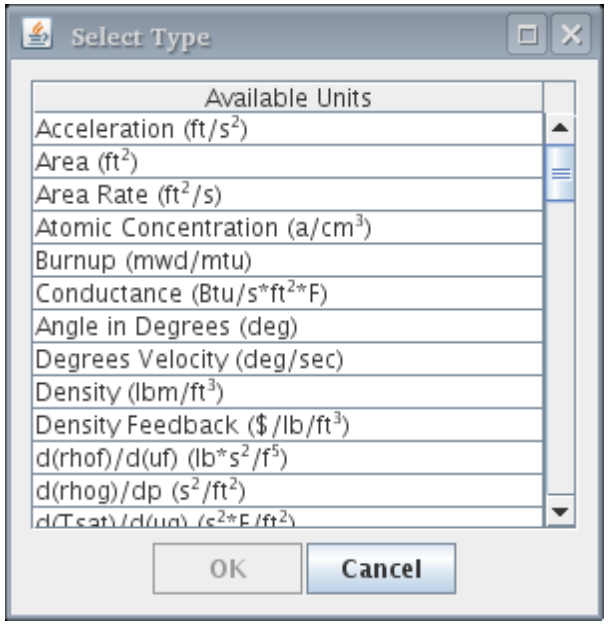
User Defined Numerics

- Variables may be used in place of entering an explicit value for component data.
- Python, Matlab or MathCAD functions can be implemented to calculate variable values.
- Variables are either Interactive or Calculated.
  - Interactive variable values are entered directly.
  - Only interactive variables may be used as independent variables in job streams.
  - Calculated variables are assigned values by user defined functions.
- Variable Types:
  Boolean, Integer, Real, String, Table
- Real variables have engineering units (Area, Length, etc.).
- Variables can only be used by a property that has the same units.



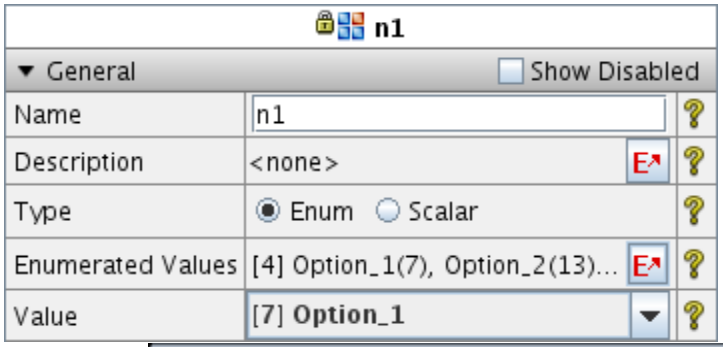
Real Variables

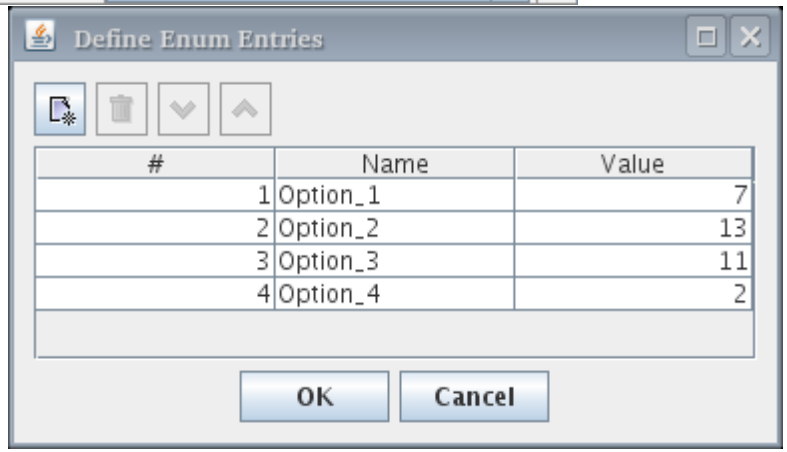
- Type defines units, not editable after instantiation
- Values converted automatically when model engineering units change between SI and British.
- Interactive
  - cannot be modified by functions
  - no initial value
  - can be Edited in a View
- Non-interactive
  - values can be modified by functions
  - value reset to initial value before functions are evaluated

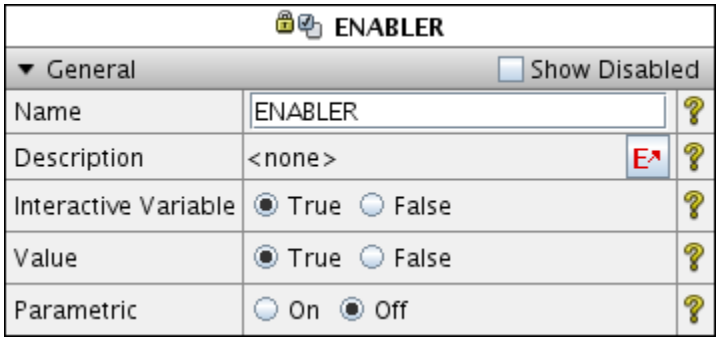


Integer & Boolean Variables

- Integers
  - Scalar or Enumerated
  - Enumerated: Name, Value, Current Value
- Booleans
  - Useful for controlling the execution of functions







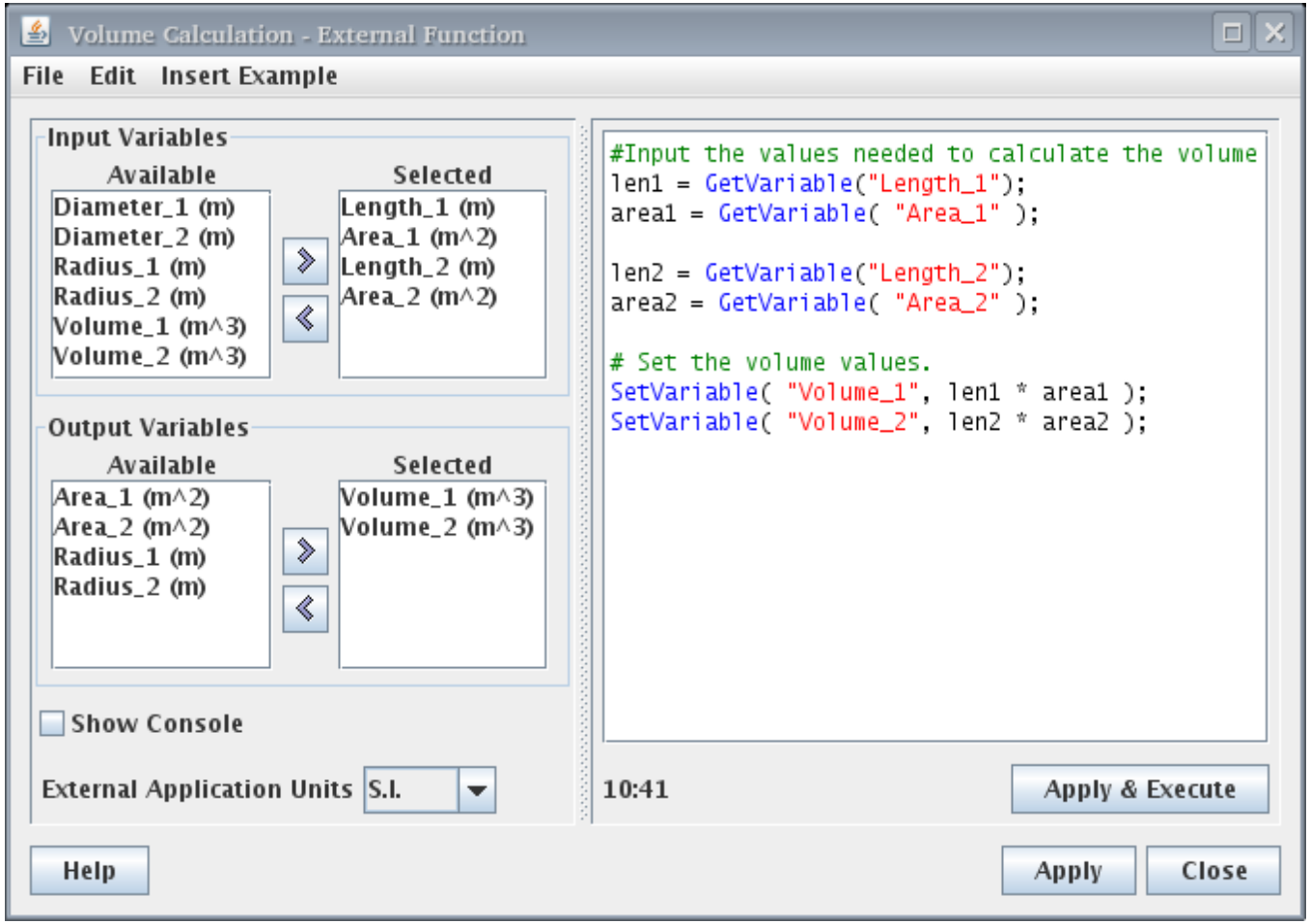
User-Defined Functions

- Python, Matlab, Mathcad
- Operate on Numerics Variables
  - Input: Interactive and Non-Interactive Variables
  - Output: Non-Interactive Variables
- Customizable execution order
- Enabling Booleans control execution
- Python- function source embedded in .med file
- Matlab/Mathcad- references to .m/.xmcd files
- Application Options- units, console, application window (Mathcad only)

Python/UDN Interface

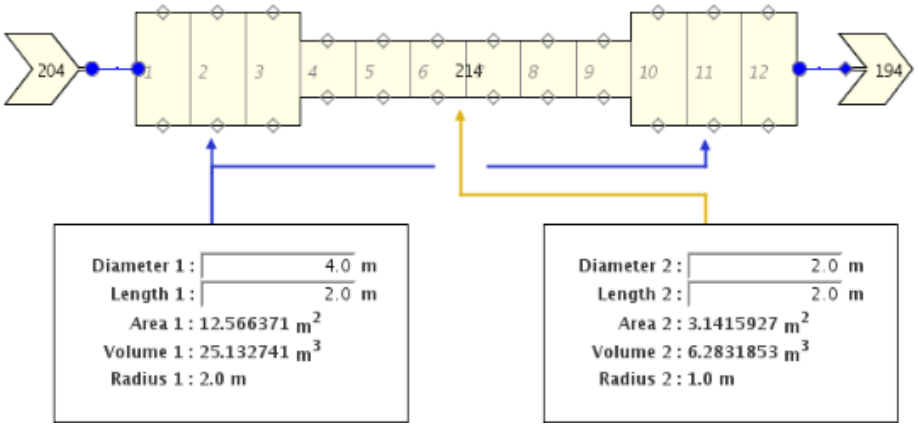
- Python code interpreted via Jython
- Python function source stored in External Function editor
- Input, output variable lists inferred upon function execution
- Custom Python module for accessing and modifying variables
- Python Interface Methods
  - GetVariable(“name”)
  - SetVariable(“name”, value)
  - GetTable(“name”)
    - GetRowCount()
    - GetColumnCount()
    - GetValueAt(row, column)
    - SetValueAt(row, column, value)

Sample Python Function



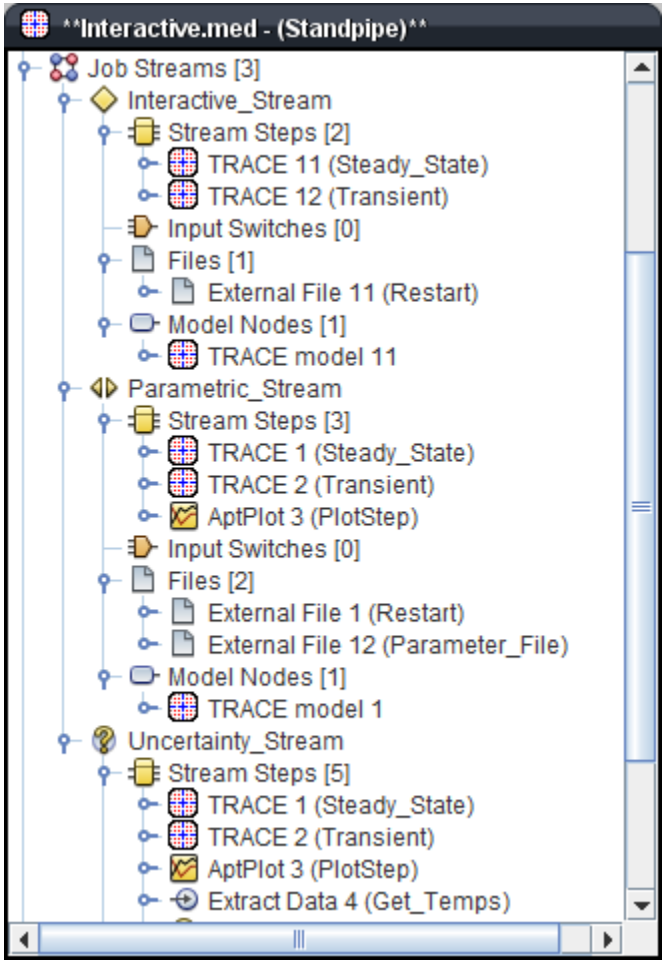
Variables in Views

- Booleans, Integers, Reals
- Drag and drop, or use “Add to View” right-click menu item
- Interactive variables can be set to editable
- Changing a value executes User-Defined Functions
- Values updated after function execution



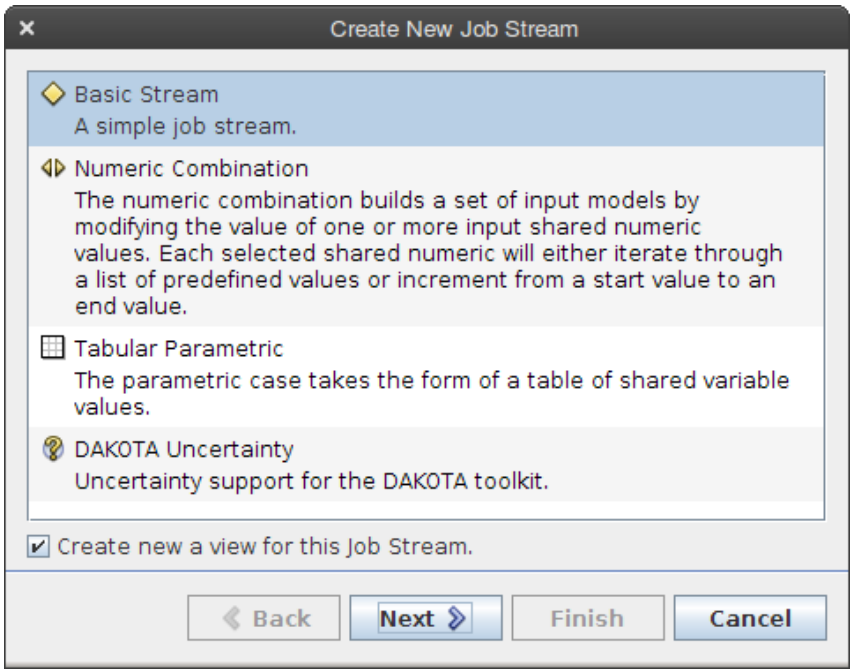
Job Streams

- Job Streams define how a series of interrelated applications should be executed.
- Graphically construct sequences of Job steps.
- A Model can contain any number of Job Streams.
- Parametric Support through Numerics and File connections.
- Parametric Fan-Out and Fan-In.



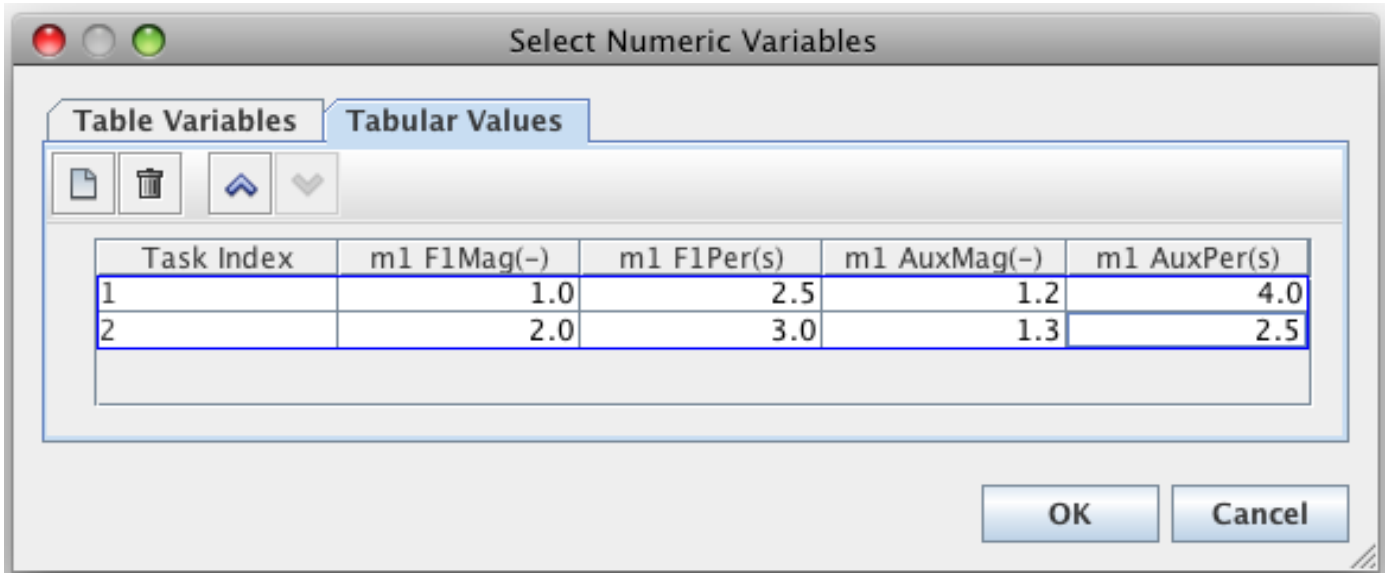
Job Stream Types

- Basic Job Stream
  - Non-Parametric cases
- Numeric Combination
  - Generate a set of parametric cases by specifying one or more interactive user defined numeric variables.
  - A set of values is specified for each variable.
  - Each combination of values corresponds to a parametric case.
- Tabular Parametric
  - Create a table of UDN variables
  - Each row corresponds to a parametric case.
- Dakota Uncertainty



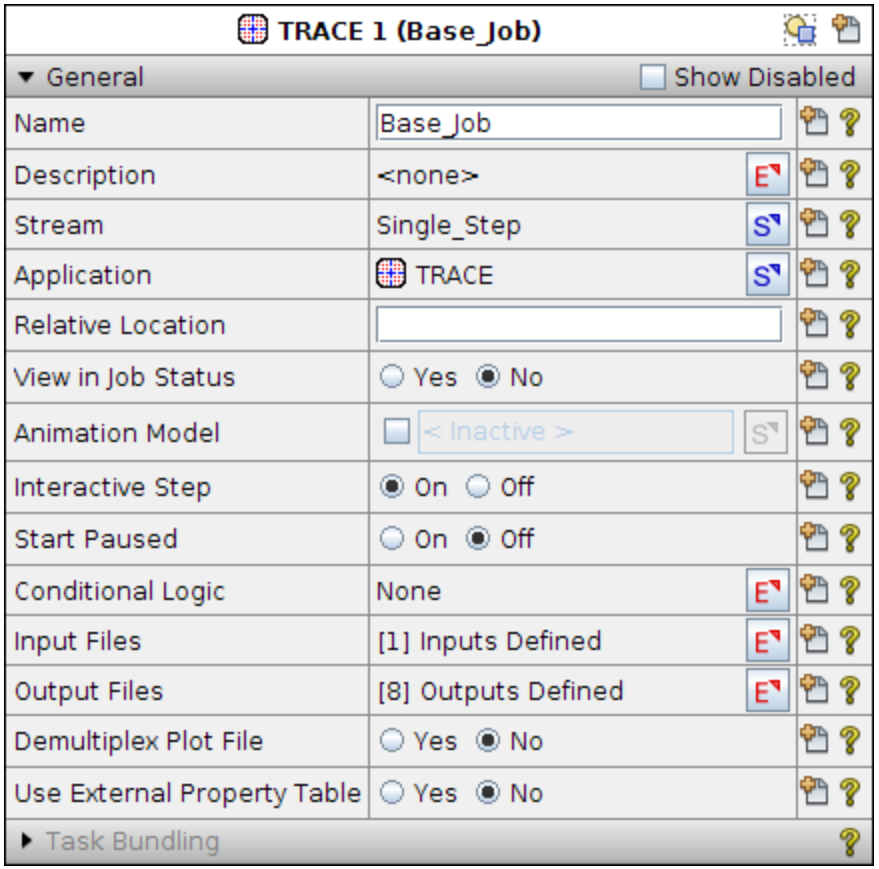
Tabular Parametric

- An explicit table of variables.
- Each row corresponds to a parametric case.
- Each independent variable must have a value in every row.
- The order of the rows can be changed.
- Supports integer, Boolean and real variables.



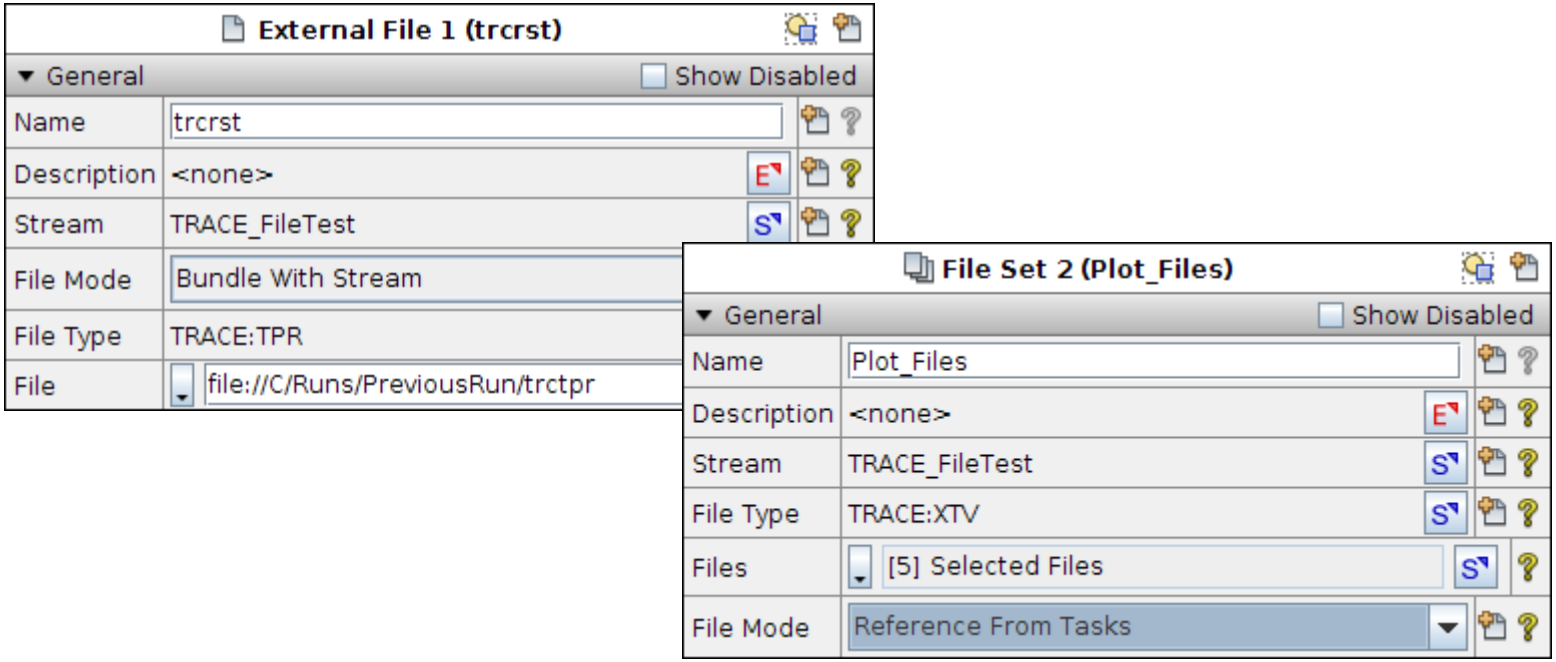
Job Streams – Job Steps

- Job Steps can include:
  - Input Pre-processing (IC retrieval, Renodalize, etc.)
  - Analysis Code Execution (TRACE, PARCS, etc...)
  - Post-Processing (Plot generation, Data Extraction, etc...)
  - Notifications (E-mail)
- Application Outputs can be archived after execution.
- Conditional Logic can be used to determine if a step should be included in the stream submission.



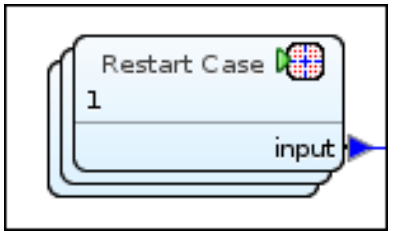
Job Streams – External Files

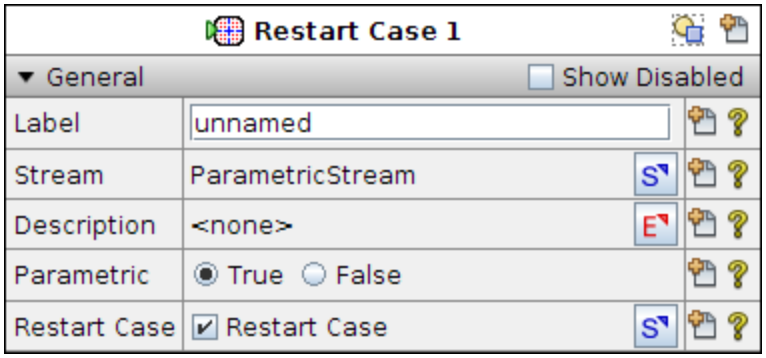
- Represent one or more files that will be included in the Stream.
- Each file has an explicit file type that determines where it can be used in a Job Stream. e.g. TRACE:XTV, PARCS:Output
- Files can be bundled with the stream, retrieved as they are needed during execution, or directly referenced.



Job Streams – Model Nodes

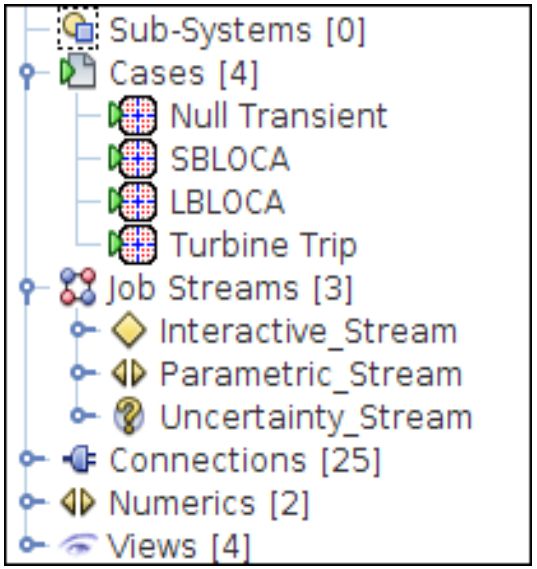
- Model Nodes represent a model in a Job Stream.
- May refer to a Restart Case.
- A Job Stream can contain multiple Model Nodes.
- Can be Parametric if the Stream Type allows.
- Only one Model Node can be Parametric within a Job Stream.

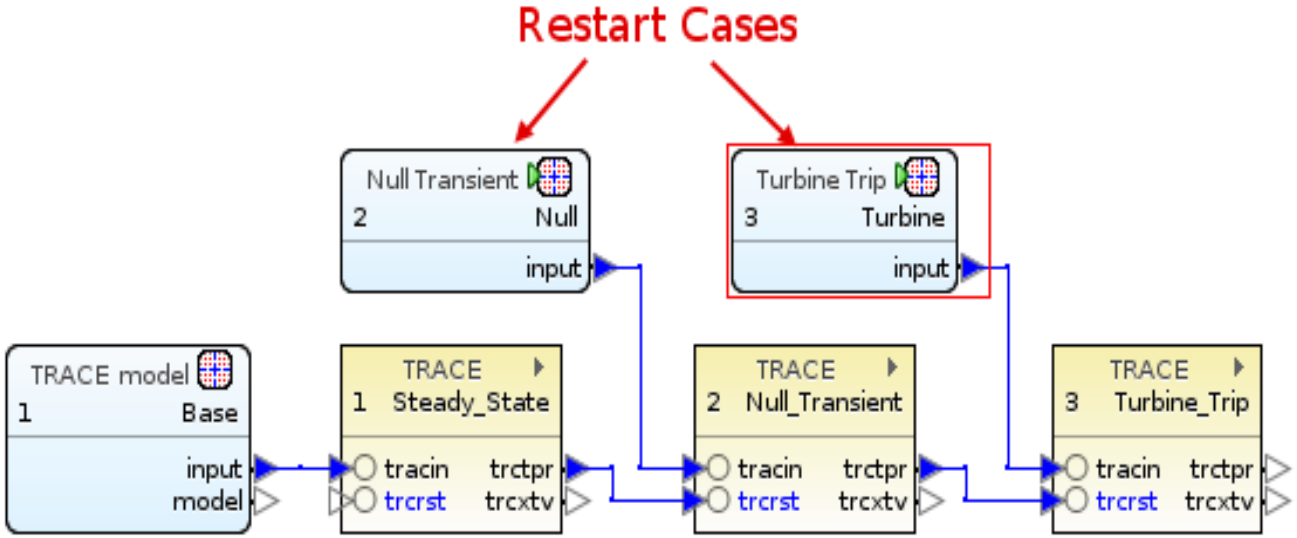




Restart Cases

- Replaces 'Editing Restart' Mode.
- A model can have any number of Restart Cases.
- Cases may be edited either graphically or by directly modifying the ASCII input.
- Base Model remains unaffected.
- Cases can be imported from and exported to ASCII files.
- Cases can be used directly in Job Streams.

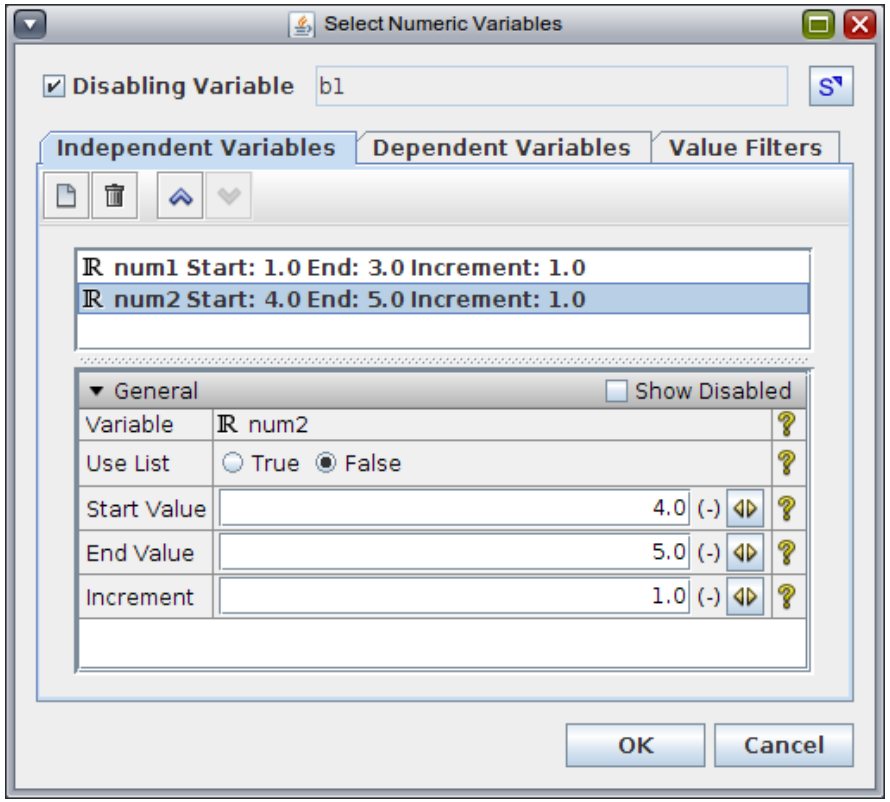




Parametric Job Streams

- Submit Sets of Calculations
  - Selected Variables Modified Parametrically
- User Defined Numerics and/or Global Variables
- Multiple Types Supported:
  - Numeric Combination
  - Tabular Parametric
- Cases May be Filtered
  - Conditional Logic
  - Explicitly Included/Excluded
- Engineering Template
  - Global Variables
  - Model References

Numeric Combination



- Builds a combination of values from each of the selected independent variables.
- Either an array of values or a start, end and increment set is defined for each variable.
- Each combination of values corresponds to a parametric case.
  - For example: x has 3 values, y has 7 values, the parametric set will contain 21 cases.

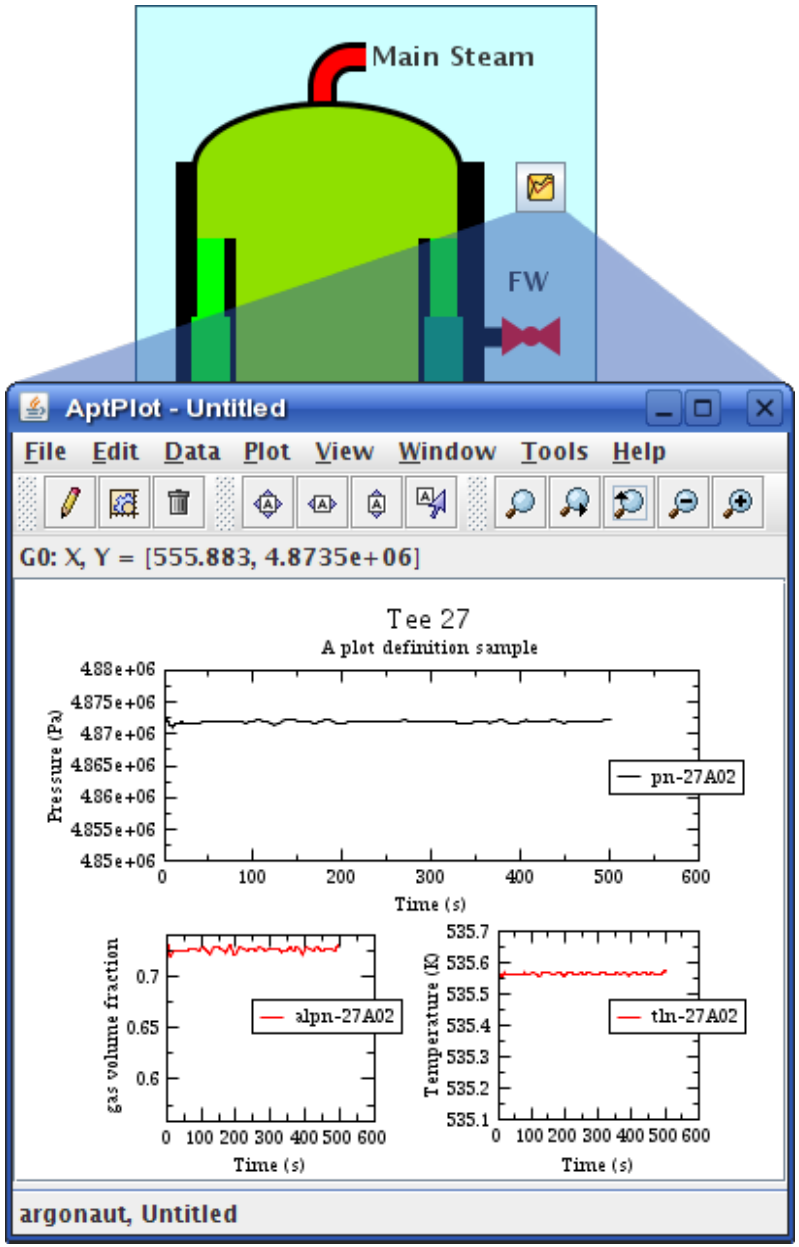
Animation Models

- 2D & 3D Visualization of the results provides insight into the model.
- Interactive or Replay Modes.
- The views can be built from the 2D views of the pre-processor model
- Can include data from multiple runs, different codes or experimental results in the same animation.
  - The Time data obtained from the master run
  - The slave run data is interpolated to match the master time steps
- Color Maps (aka Data Ranges) map data values.
- Python Data Sources – Custom Data Source
- Easily Add Customized Display Beans at Runtime

Interactive Runtime Controls

- Rewind – Back to the beginning of the calculation
- Fast forward – To the end of the calculation, or the last written timestep
- Step Forward/Backward – Advance or rewind by a fixed number of timesteps
- Replay completed or interactive calculations
- Replay proportional to realtime
- Pause or continue calculation during Replay
- Playback Controls, Status & Time Slider Beans

Plot Definitions



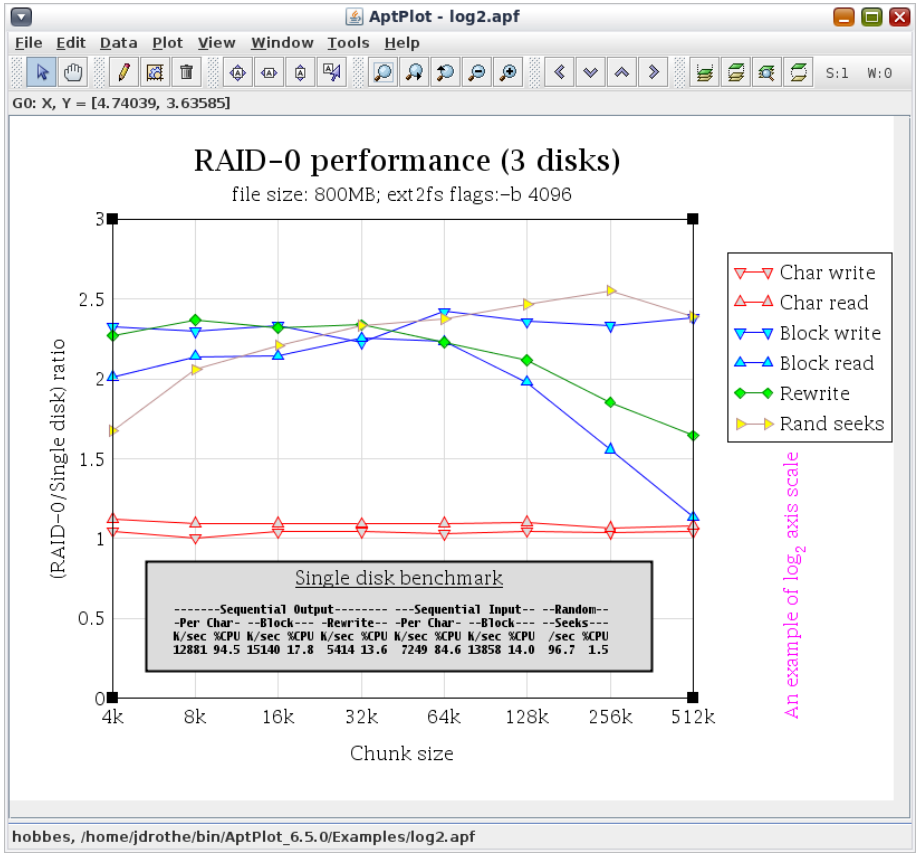
- Persistent plot definitions saved with an Animation Model
- Defines formatting with a parameter file saved from AptPlot
- Maps channels directly to Graphs and Sets
- Can add custom batch commands to tweak the graph
- Drag & Drop Plot Definitions onto Views to create plot launchers.

Engineering Templates

- Used to construct complex Job Streams using multiple source models.
  - i.e. TRACE - PARCS → FRAPCON/FRAPTRAN
- Create input templates to constrain modifications to one or more source models.
- Source Models are queried for available numerics variables. These variables are then available to create 2D views which serve as template forms.
- Underlying Models are manipulated through User Defined Numerics.
- 'Global' engineering template variables can be mapped to UDNs of underlying models.

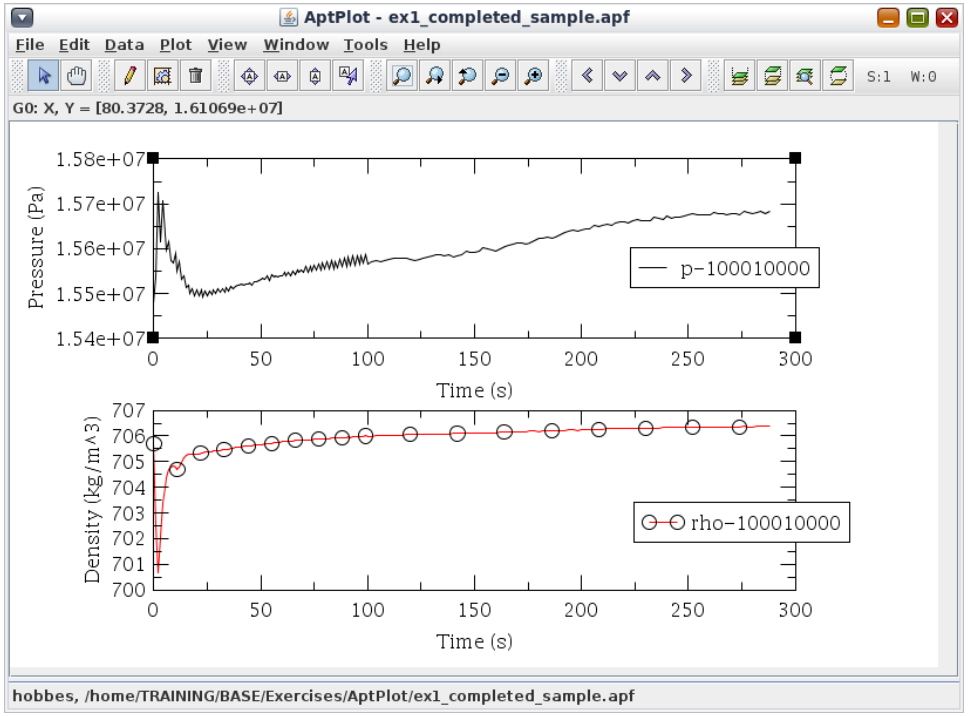
AptPlot

- Pure- Java WYSIWYG Plotting Application
- AptPlot is a descendant of AcGrace & Xmgr5
- AcGrace:
  - Written in C on top of Motif
  - Requires an X11 server to run
  - Requires Cygwin in Windows
- Creates Publishing-Quality plots
- Supports Several output formats:
  - PDF, JPG, PNG, EMF, TIFF, PostScript
- Analysis Code Support Plug-in (ACS)



AptPlot Basics

- An AptPlot canvas is a single plot
- Plot – The root AptPlot object
- Graph – A single graph
  - Plots may consist of one or more graphs.
- Data Set – A single element of the graph
  - Graphs may consist of one or more data sets.
- Scalar & Vector Variables
  - Used for Calculations
- Text, Line, Rectangular, and Circular annotations are available



AptPlot: Plots

- All graphs and annotations are placed on the plot
- Plot objects are placed with View Coordinates
- Plots allows basic global formatting: background color, size, etc.

AptPlot: Graphs

- Each graph is a view of its data sets, axes, legend, etc.
- Objects within the graph are placed using World Coordinates
- Diverse graph-wide formatting: background color, frame style, etc.
- Provides several formats for data set interpretation and display
  - XY line, Bar graph, Pie chart, Polar graph, etc.
- Axis scaling: linear and logarithmic

AptPlot: Data Sets

- Two or more columns of data, as dictated by graph and set type
- Type can be specified, limited by the graph type
- XY for basic data values
- XYDY for basic with vertical error bars
- XYDYDY for basic with vertical error bars specified for both directions.
- Each set has specific formatting
  - Line color, width, and style
  - Symbol type, line, color, fill
  - Annotated values and their formatting
  - Error bars placement, color, line format

AptPlot: Scripting

- Batch commands can control all aspects of a plot: formatting and data
- Parameter & Plot Files Use Batch Command Syntax
- Simple command syntax
  - case insensitive “declaration” style commands
  - examples:
    - TITLE “title” - set the current graph's title
    - COPY G0.S1 TO G0.S2 – copy the data in set1 to set2
- Print plots directly to several supported formats
  - HARDCOPY DEVICE “PDF” -set the output image type to PDF
  - PRINT TO “filepath” - print the PDF to the given location
- Read data from and write data to ASCII files
  - READ “filepath” - read the data at the given location
  - WRITE set FILE “filepath” - write set data to the given

AptPlot: Scripting

- Advanced data manipulation can be invoked by batch commands
  - RUNAVG(set, n) – running average of set using n points
  - HISTOGRAMS(set, bins, cumulative, normalize) – calculates a histogram of the set
- Supports headless mode for batch execution on a headless server
- Equation Interpreter: additional functionality over standard commands
  - Identified from normal commands as: CALC “expression”
  - Create user-defined scalar and vector data
  - Additional convenience functions
  - Automatic vector interpolation across all operations
  - Ties directly into the ACS plug-in

Analysis Code Support (ACS) plug-in

- Allows reading analysis-code plot-file data directly
  - Select code type, file type, and file.
  - Supports demultiplexed (demux) plot files
  - Demux files are optimized for plotting
  - Installation includes demultiplexers for supported codes
- Ties directly into scripting
  - Open command makes the plot-file available
    - TRAC DEMUX “filepath”
  - Read command to mark which channels to read and retrieve them
    - TREAD “pn-12A01”
    - TREAD DONE
  - Ties directly into Equation Interpreter:
    - CALC “G0.S0 = t0_pn-12A01”
- Currently supports the following plot files:
  - TRAC/TRACE: XTV, TRAC-B: TRCGRF, RELAP5: RSTPLT, PIB, & Strip, MELCOR, PARCS, COBRA, CONTAIN, GOTHIC, EXTDATA, NRC Databank

SNAP Diagnostic Output

- SNAP Applications write messages to screen and log files that can help diagnose issues.
  - Model Editor – me.screen
  - Job Status – js.screen
  - Calculation Server – cs.screen
- SNAP user preferences and screen files are stored in:
  - Windows 7 or Vista:
    C:\Users\<username>\.snap\2.0\
  - Windows XP:
    C:\Documents and Settings\<username>\.snap\2.0\
  - Unix/Linux:
    $HOME/.snap/2.0/
- The Calculation Server also writes a log file to:
  <home>\.snap\2.0\log\calculation_server.log
- Job Streams and Tasks write log files in their folders.
  - Job Stream – <stream name>.streamlog
  - Task – <task name>.tasklog