

Getting Started with the RSF Toolkit

Martin R. Holmer, Policy Simulation Group

September 2010

The RSF Toolkit automates the creation of a new PSG models run using a Run Specification File (RSF) and it also provides capabilities for executing the run and analyzing run results. A run can execute in any mode of operation available in the PSG models. The Toolkit provides an easy way to specify a run without having to directly manipulate the PSG models input databases.

An RSF is a text file describing a run of the PSG models. An RSF usually contains two sections: a preamble that describes which mode of operation is being used, and a body that describes the assumptions and reform proposal being simulated. A reform proposal is composed of reform provisions, each one of which specifies how pension or social security policy parameters differ from their current-law values.

The RSF Toolkit provides capabilities for automatically building an RSF by answering questions in an interview. The resulting RSF may be hand edited in the Toolkit, and then used to create automatically all the input database table rows required to specify the run. These RSF-build and run-create capabilities automate the most difficult tasks involved in using the PSG models for pension and social security analysis.

Contents

It is assumed that the reader of this document has already read “Getting Started with the PSG Models” (<http://www.polsim.com/psghelp.pdf>).

This short primer contains the following sections:

Run Specification File Example (page 2) provides an example of an RSF to introduce the syntax of the run specification language.

RSF Toolkit Capabilities (page 4) provides a brief description of each individual Toolkit capability.

RSF Toolkit Operations (page 11) provides a number of descriptions about how to accomplish common tasks that require using a sequence of Toolkit capabilities. This section consists of answers to a series of “How do I . . . ?” questions.

RSF Language and Scope (page 16) provides a detailed language definition and a list of the input database tables that can be manipulated using an RSF.

Run Specification File Example

The run specification file (RSF) is at the center of the process of automatically creating new runs. An RSF contains simple instructions about how a new run is the same as an existing run except for a list of differences. This is the most common way policy analysts describe a reform: they say a reform is like current-law policy except that it differs in a number of ways. Each policy difference is called a reform provision, and the reform proposal is composed of one or more reform provisions. The simple RSF instructions, which will be described below, formalize this way of describing a reform run.

The process of automatically creating a new run involves two steps. The first step is to *build* an RSF; the second step is to *use* the RSF to create all the database table rows for the new run. The RSF Toolkit automates both steps. The Toolkit conducts an interview to determine the desired mode of operation and the desired policy reform, using the interview responses to build an RSF. The RSF is built in an editor where it can be edited, printed, saved, and manipulated in other ways.

Consider the following example. Suppose we want to use the CBA mode and 2010TR assumptions to estimate the 75-year solvency effects of a reform with two provisions: (a) gradually raising the DI payroll tax rate on earnings below the maximum taxable earnings level from 1.8 in 2013 to 2.1 percent in 2016, and (b) gradually raising the OASI payroll tax rate on earnings below the maximum taxable earnings level from 10.6 in 2013 to 12.3 percent in 2020. Here is the RSF built by the Toolkit interview:

```
## RUN.id ?
## SSASIM CBA mode run:

create ssasim run same as run 10 except:
table RUN field n_test_yrs is 75
table RUN field num_years is 81
table RUN field scenarios is 1
# OASDI reform provisions:

# Change payroll tax on earnings below maximum taxable earnings
# Youngest-age-group DI tax rate on earnings below MTE
table TAXR_DI field ptax_rate is same thru 2013 and 2.1 for 2016
# Youngest-age-group OASI tax rate on earnings below MTE
table TAXR_OI field ptax_rate is same thru 2013 and 12.3 for 2020
```

The example RSF begins with a comment where the RUN.id value needs to replace the question mark. The second line is a comment that identifies the mode of model operation and start of the RSF preamble. Any part of a line beginning with and following a # character is considered a comment, which is ignored in the RSF-use step. The comment character # can appear anywhere on a line, but any text to the left of it is not a comment and will be read in the RSF-use step.

The line in the example RSF that begins with `create` and ends with `except:` is a create statement that specifies the model being used and the existing run that serves as the base run from which to create the new run. Run 10 is an CBA mode

benchmark run that assumes 2010TR demographic and economic assumptions and represents current-law policy. The remaining lines in the example RSF are except statements that specify exactly how the new run differs from the base run. Notice that these except statements start by identifying a table and field, and then specify the value of the field. The way the value of a field is specified depends on whether the table is dynamic or static. A table is dynamic if it contains a secondary key that indicates year or age, meaning that the table's parameters can change by year or by age. A static table contains no such secondary key.

The RUN table is static, and so in the example RSF, a simple `is value` construct is used to specify the single value of the parameter `n_test_yrs` (the number of years in the solvency test period) and `num_years` (the total number of simulation years). The Build interview asks for the number of years in the solvency test period (offering 75 as the default value) and then calculates the required number of simulation years. These two RUN-table except statements complete the RSF preamble.

The rest of the file is the RSF body that describes reform provisions or assumption changes. The first provision calls for a gradual rise in the DI payroll tax rate over a three year period. After asking in the interview when the rise starts and stops and how high the rate goes, the Toolkit builds the comments and except statement starting with `table TAXR_DI field ptax_rate`. The TAXR_DI table is dynamic, and therefore, the except statement is more complicated after the `is` keyword. The `same thru 2013` clause is used to specify the period over which the policy parameter is unchanged from the base run. After the `and` keyword, the `2.1 for 2016` clause indicates the timing and magnitude of the tax rate rise (up from the 1.8 current-law value in the base run by 0.1 percent per year).

The second provision calls for a gradual rise in the OASI payroll tax rate over a seventeen year period from 10.8 to 12.3 percent. The build interview generates `table TAXR_OI field ptax_rate is same thru 2013 and 12.3 for 2016`. And then, after the build interview is finished, the 2016 is edited to be 2020 to make the rate increase more gradual.

It is essential to remember that the PSG models perform automatic linear *interpolation* and *extrapolation* of parameters in dynamic tables. This useful feature eliminates the need, in this example, to specify explicitly that the DI payroll tax rate will be 1.9 percent in 2014 and 2.0 in 2015, and that the tax rate will be at 2.1 percent in all years after 2016. For the same reason, there is no need to calculate OASI payroll tax rates for years between 2013 and 2020.

The Toolkit automatically uses an RSF to create all the table rows in the input database needed to specify a run after a new RUN.id value is specified on the first line of the RSF.

A useful approach to getting started with the RSF Toolkit is to finish reading this primer and then use the Toolkit to build and use this example RSF. After executing this run, be sure to visualize the output statistics using the TFS Toolkit or view the raw output statistics using the Analyze menu.

RSF Toolkit Capabilities

The capabilities of the RSF Toolkit are activated by pressing one of several buttons or pulling down one of several menus. These buttons/menus appear at the top of the run specification editor that is the main window of the Toolkit. If the mouse cursor is held over a button for more than a couple of seconds, a short help message will appear. The last item on each menu displays information about the capabilities associated with each item on that menu.

Both menus and buttons can be activated either by clicking with the left mouse button or by pressing the key of the underlined character in the menu or button name while holding down the Alt key.

Dialog boxes that appear can be dismissed by clicking with the left mouse button on the x icon in the upper-right-hand corner of the box (unless the dialog demands an answer to its question) or by pressing the Esc key.

This section of the primer lists the capabilities associated with each button and menu item. The next section of the primer describes how to combine these individual capabilities into more complex operations.

The menus/buttons are grouped, via four red bars, into three sections corresponding to the three steps in manipulating model runs in the policy analysis cycle: specify, execute, analyze. The menus and buttons used in the run specification step appear on the left.

File menu. The File menu presents a choice of several actions that can be applied to one or more editable files. In addition to run specification files (*.rsf), four other types of files (used in post-simulation processing of raw model output) can be manipulated with the File menu items: Windows batch scripts (*.bat files), Tcl programs (*.tcl files), AWK programs (*.awk files), and post-simulation tabulation results (*.res files).

Show RUN.id in all RSF on disk. Shows all the run specification file names and the included RUN.id value for each file. The list can be ordered either by RUN.id or by RSF name.

Contrast two files on disk. Shows the differences between two files using the kdiff3 utility. The kdiff3 utility can be configured to show the two files side-by-side or top-and-bottom using the Toggle Split Orientation item on the Window menu, and to display the lines with or without word wrap using the Word Wrap Diff Windows item on the Diffview menu. Move from one group of lines with differences to another using the up and down arrowheads on the tool bar or by clicking on the blue bars that mark lines with differences. Exit the kdiff3 utility by clicking on the x icon at the upper-right corner of its window or by using the Quit item on the File menu.

Search files on disk for keywords. Shows file name, line number, and line contents for each line in all files of the file type shown in the RSF Toolkit editor that contain the specified keywords. The keyword search is case-insensitive. The search starts with the left-most keyword and then searches the results of that first search using the second keyword, etc.

Find & replace phrase in files on disk. Asks for a find phrase and a replace phrase and a list of files on disk, and then uses this information to substitute the replace phrase for each occurrence of the find phrase in every file on the specified list. There is no undo of this find-replace action, so it would be prudent to do a backup before starting to gain experience with this feature. Note: the list of files is specified in a file open dialog box, where multiple files are selected using standard Windows methods for multiple selection in a dialog box: holding down the shift key selects a contiguous group of files; holding down the control (Ctrl) key selects multiple non-contiguous files. To reduce typing, select in the run specification editor the phrase to find and it is copied into the find-phrase box, or do a text copy and paste into the box.

Print file in editor via Notepad (Ctrl+P). Loads the file being viewed in the editor into the Windows Notepad where it can be printed. It is important to print an RSF using 10pt Courier New as the Notepad font.

Rename file on disk. Renames a file that already exists on the disk.

Delete file on disk. Deletes a file in the current folder from the disk. And, in addition, if a run specification file, deletes runs from model databases that were specified using the selected RSF, and deletes all model output files produced by these specified model runs. There is no undo of this delete action, so be sure you want to erase all these files before confirming the delete action.

Backup files to psgYYMMDD.zip. Copies all files related to the PSG models to a zip file with today's date and places zip file in the specified storage folder. Creates an backup archive file containing all PSG model files (*.rsf, *.tfs, *.eda, *.h2d, *.eci, *.awk, *.sql, *.tcl, and *.bat). The backup archive file is named psgYYMMDD.zip, where YY denotes the year, MM the month, and DD denotes the day the backup file is created. It is prudent to make a backup at least at the end of every day you use the PSG models. And it is best to put the backup archive file in a folder/directory that is safe (such as a network drive that is backed up to tape every night). The default backup directory is the parent folder/directory of the SSASIM installation folder/directory, but you can change that by editing the tkprefs file as described above in the 'Changing toolkit preferences' section.

Change database folder. Allows change in database folder, which is useful only if you maintain more than one input database folder.

Exit RSF Toolkit. Quits the RSF Toolkit after asking if unsaved contents of the RSF editor should be saved. The x icon at the upper-right corner of the Toolkit window exits in the same way.

Open button. Reads a file from disk and places its contents in the editor. The editor can open, edit, and save many file types including RSF (*.rsf files), AWK programs (*.awk files), Windows batch scripts (*.bat files), Tcl programs (*.tcl files), text results (*.res files), and all text files (*. * files).

Save button. Writes the contents of the editor to the disk using the file name shown in the Toolkit title bar.

SaveAs button. Writes the contents of the editor to a file on the disk using a specified file name.

Clear button. Erases the contents of the editor. If the editor contents need saving, an opportunity to save will be given.

Build button. Starts a Build interview that asks questions about what kind of run is desired and constructs an RSF as interview responses are supplied. The first question of any Build interview is about which of the two kinds of interview is desired.

Build reform-policy Run Spec up from current-law policy. This is the appropriate choice when starting from the beginning in specifying a reform proposal. A run characterizing current-law policy will be used as the base run used when this choice is made.

Add reform provision(s) to the current Run Spec. This is the appropriate choice when an RSF with a preamble and possibly some reform provisions is present in the editor and the objective is to add more reform provisions. The preamble interview is skipped when this choice is made.

Reform Provisions. Once the preamble questions have been answered, the Build interview moves into a reform specification phase. A second editor-like window appears with a Kind button, Search menu, Add button, Quit button, and a Help menu that explains what each button/menu does and provides the capability of printing in Windows Notepad the complete contents of the Reform Provisions window.

Edit menu. The Edit menu presents a choice of several editing actions that can be applied to the text in the editor. Most of the items on the edit menu can all be invoked using the keyboard (that is, without posting the menu); the hot key for each item is shown on the edit menu.

Undo (Ctrl+Z). Undoes editor (and just editor) actions. There is no limit to the number of actions that can be undone, but saving an RSF to disk eliminates the list of editor actions that can be undone.

Redo (Ctrl+Y). Undoes the previous undo action.

Cut (Ctrl+X). Removes the selected text in the editor and places it on the Windows clipboard.

Copy (Ctrl+C). Copies the selected text in the editor and places it on the Windows clipboard.

Paste (Ctrl+V). Copies the contents of the Windows clipboard into the editor at the cursor location.

Top (Ctrl+Home). Moves the cursor to the first line of the editor.

Bottom (Ctrl+End). Moves the cursor to the last line of the editor.

Find and find again (Ctrl+F). Prompts for keyword used in a case-insensitive search forward from the cursor location. The up and down arrow keys scroll through prior search keywords. Press Ctrl+F to search forward again and Esc to terminate the search.

Find terminate (Esc). Stops a forward search process.

Select all. Selects all the text in the editor.

Comment selected region (Ctrl+=). Inserts one comment character (#) at the beginning of each line in the selected (that is, highlighted) text region.

Uncomment selected region (Ctrl+-). Removes one comment character (#) at the beginning of each line in the selected (that is, highlighted) text region whenever a line starts with a comment character.

Line number (Ctrl+L). Shows the editor line number of the line on which the cursor is located.

Show base run table field values (Alt+V). Shows, in the Windows Notepad, values of all fields in the selected table for the base run. The “base run” is the model run specified on the RSF create command just above the selected table. The “selected table” is defined as the table named in the editor line where the cursor is located. Being named means that the first word on the line is table and the second word is the name of the table. If the cursor is located on a line that does not start with table, nothing is shown in the Windows Notepad. IMPORTANT NOTE: this capability is most easily activated using the Alt+V keystroke. If the base run uses many rows in the table, be sure to turn the Windows Notepad word-wrap feature off to view the results.

Show table field documentation (Alt+D). Shows, in your default browser, documentation for the table field on the editor line where the cursor is located. If the line has both a valid table and a valid field, the browser displays the table documentation with the page focus on the field. If the table is valid but the field name is missing or invalid, the browser displays the table documentation with the focus at the top of the page. If the table name on the line is missing or invalid, the browser displays nothing. IMPORTANT NOTE: this capability is most easily activated using the Alt+D keystroke.

Use button with run specification in editor. If the run specification in the editor has not been given a name and saved to disk, then clicking the Use button prompts for an RSF name.

New Run Number. Before saving the contents of the editor in an RSF, the question mark on the first line of the RSF must be replaced by a number that specifies the RUN.id value to be assigned the new run specified in the RSF.

Parameter Validity Check. After creating the new run in the input database, the Toolkit checks all the parameter values used in the new run to ensure that they all have valid values. If there is a syntax error in the RSF or if an input parameter value is not in the valid range, a message will be generated indicating the problem. Simply fix the problem in the editor and press the Use button again.

QUEUE Table Insertion. The final step in the use process is an optional insert of the new run number in the SSASIM QUEUE table. If you don't insert the run number in the QUEUE table, this can be done later using an item on the Execute menu.

Use button without run specification in editor. If the editor is empty (say, after pressing the Clear button) when pressing the Use button, a dialog starts that allows the selection of multiple run specification files for use. Multiple files are selected using standard Windows methods for multiple selection in a dialog box: holding down the shift key selects a contiguous group of files; holding down the control (Ctrl) key selects multiple non-contiguous files.

Execute menu. The Execute menu presents a choice of several ways to initiate run execution or to test the validity of a model installation with a benchmark run. It also provides access to the run log file and the error log file.

Single run in editor (Ctrl+G). Perhaps the most common sequence of operations is saving to disk the run specification file being edited, using that run specification file, inserting its RUN.id into an empty QUEUE table, and executing that run without directing its screen output to the run.log file. The go key, Ctrl+G, activates this sequence of operations, providing an alternative to clicking first on the Use button and then selecting the first item on the Execute menu (see below).

Runs after QUEUE view. Starts window that provides a view of the contents of the SSASIM QUEUE table, as well as the capabilities of inserting database runs into the QUEUE table, deleting run numbers from the QUEUE table, viewing the RUN.notes field of any run in the QUEUE table, and starting the model executing the runs in the QUEUE table.

Run numbers are inserted into the QUEUE table just before the run number that is highlighted in the list. Also, notice that a notional run called Q-END represents the end of the QUEUE table list. So, if you want to add a run number at the end of the QUEUE table list, select the Q-END and then insert the run number by clicking on the QUEUE Insert button. The QUEUE Delete button removes only the selected run number.

Runs on Amazon Web Services. Starts the AWS Toolkit that uses the Linux version of the PSG models to provide the distributed processing of multiple-scenario runs via Amazon Web Services (AWS) EC2 compute servers and S3 data storage.

1 – CBA mode benchmark run. Starts the SSASIM CBA mode benchmark run executing.

3 – OLC mode benchmark run. Starts the SSASIM OLC mode benchmark run executing. Do this only if both the PENSIM and GEMINI (standard version) add-ons to SSASIM are installed.

5 – ESP mode benchmark run. Starts the PENSIM ESP mode benchmark run executing. Do this only if the PENSIM add-on to SSASIM is installed.

7 – RCS mode benchmark run. Starts the GEMINI RCS mode benchmark run executing. Do this only if both the PENSIM and GEMINI (standard version) add-ons to SSASIM are installed.

9 – ESP+ mode benchmark run. Starts the PENSIM ESP mode benchmark run executing with pension results being passed through to the GEMINI output files. Do this only if both the PENSIM and GEMINI (educational or standard

version) add-ons to SSASIM are installed.

View run.log file. Shows the contents of the run.log file in Windows Notepad.

View error.log file. Shows the contents of the error.log file in Windows Notepad.

Erase all log files. Deletes all files that end in .log in the current SSASIM database folder, and in the corresponding PENSIM and GEMINI database folders, if they exist.

Analyze menu. The Analyze menu presents a choice of several ways to analyze the output files generated by executed runs.

OASDI solvency output with TFS Toolkit. Starts the Trust Fund Solvency Toolkit, which provides capabilities for visualizing a variety of social security solvency statistics generated by SSASIM operating in either the CBA or OLC mode with either deterministic (single scenario) or stochastic (multiple scenarios) output results. For more information, use the Help menu in the TFS Toolkit.

Distributional output with EDA Toolkit. Starts the Exploratory Data Analysis Toolkit, which provides capabilities for visualizing the distribution of social security and pension statistics contained in a number of GEMINI output files. For more information, use the Help menu in the EDA Toolkit.

Individual OASDI output with ECI Toolkit. Starts the Exemplary Cohort Individual Toolkit, which provides capabilities for specifying exemplary individuals, extracting social security statistics from ECI mode output files (.cNf and .cNm), and presenting the extracted information in an Excel table. For more information, use the Help menu in the ECI Toolkit.

Raw SSASIM output statistics. Shows a dialog box that provides a view of any SSASIM output file. The bottom two buttons provide the ability to compare the contents of the same type of output file generated by two different SSASIM runs. The differences between the two output files are displayed side-by-side with color highlighting of differences in output statistics.

Output from Windows command prompt. Starts a command prompt window in the database folder of the selected model. The output analyzer programs documented in the next menu item are available for use at the command prompt or from Windows scripts (.bat files) executed from the command prompt or as part of a post-simulation tabulation script (see documentation of SSASIM input parameter RUN.tab_script).

Output analyzer program documentation. Shows a menu of several output analyzer programs that are part of either the SSASIM, PENSIM, or GEMINI model distribution. Selecting one of these programs causes that program's documentation to be shown in Windows Notepad, where it can be read on screen or printed on paper.

Help menu. The Help menu provides access to several types of documentation and miscellaneous capabilities.

Getting Started with PSG Models. Shows the "Getting Started with the PSG Models" document in the Adobe Acrobat Reader. This document should

be read first when learning about the PSG models: SSASIM, PENSIM, and GEMINI.

Getting Started with RSF Toolkit. Shows this document in the Adobe Acrobat Reader. This document should be read second (after “Getting Started with the PSG Models”) when learning about the PSG models: SSASIM, PENSIM, and GEMINI.

Coping with RSF Toolkit errors. Explains what to do when a “Tk Application Error” box pops up. This is a box with a red circle containing a white x. Be sure to report this error as instructed.

View documentation. Provides access, via your default browser application, to all the hypertext documentation of the input parameters and output statistics and to the four PSG model manuals.

Search documentation. Provides links, via your default browser application, to the documentation files that contain a specified keyword or multiple keywords on the same line. The keyword search is not case sensitive. The search starts with the left-most keyword and then searches the results of that first search using the second keyword, etc.

Compute assumption parameters. Provides the ability to determine the values needed for use in ‘assumption trnd_mv ...’ run specification statements for the key SSASIM assumption variables that are transformed either by the log function or by the log-odds function. This ability is provided by the probtran utility, which is used to compute the values needed in ‘assumption trnd_mv ...’ run specification statements to produce desired mean and standard deviation statistics for an assumption variable from stochastic output in the SSASIM .sti output file.

View input parameters. Starts the dbview utility for the specified model. The dbview utility allows the viewing of input database table contents and showing the parameter differences between two runs in the database. If you need help on the meaning of table fields, view the documentation of input tables and parameters from the RSF Toolkit Help menu.

Delete output files. Provides the ability to delete run output files (but not the run in the input database) in all three model database folders. One or more runs can be selected for output file deletion by selecting one or more files in the dialog box. All model output files for each selected run will be deleted after a confirmation.

Changing toolkit preferences. Allows the expression of personal preferences concerning font size, default backup folder/directory, and Trustees Report year. Simply edit the tkprefs file that appears in the Notepad, save, and restart the RSF Toolkit for your preferences to take effect.

Check for new version. Compares version date of this copy of the PSG models with the newest PSG models version date via an Internet query.

RSF Toolkit version. Identifies the version date of the Toolkit.

RSF Toolkit Operations

This section of the primer provides answers to a number of “How do I ...?” questions.

How do I start the RSF Toolkit? Click on the red Tk icon located on the Windows task bar or desk top.

How do I see the whole RSF Toolkit application window? If the application window does not fit on your computer screen, your monitor is set at a very low screen resolution. Use the Windows Display control panel to increase the screen resolution to 1024 by 768. Do this by clicking on Control Panel under the Windows Start menu’s Settings item; clicking on the Display control panel; clicking on the Settings tab; and then use the screen resolution slider to get more resolution.

How do I see all the RSF Toolkit buttons? If the application window does fit on your computer screen, but you cannot see the Help menu at the right-hand side of the menu bar, it may mean that you are using larger than normal screen fonts. You can solve this problem in one of two ways. First solution is to switch back to normal size screen fonts and possibly adjust you screen resolution. Second solution is to widen the application window by dragging the lower right-hand corner of the window with the mouse (but you still might have problems reading all the message in a dialog box, for example).

How do I magnify the Toolkit on the screen? For those who have high resolution monitors and would prefer an easier to read Toolkit, edit the tkprefs file using the Help menu, save the edited file, and then restart the Toolkit. The default setting of `tk scaling 1.6` can be changed, for example, to `tk scaling 1.9` so that the type in the Toolkit menus and in the RSF editor is bigger on the screen. Note that changing the tk scaling factor will affect the appearance of all the PSG Toolkits (RSF, TFS, EDA, and ECI) and of the dbview utility.

How do I backup my work? Using the Backup item on the Files menu will archive all PSG model files into a file named psgYYMMDD.zip, where YY is the year, MM is the month, and DD is the day. Do this at the end of every day you use the models.

How do I know which reform provisions are included in the Build interview? There are at least three ways. First, you can use the Kind button and section links to move quickly to the start of each section of reform provisions. Second, you can use the Find button to do a keyword search (for example, search for `earnings test` to find the reform provision that modifies the earnings test). And third, you can simply print the complete contents of the Reform Provisions window by using the Printing item on the Help menu.

How do I specify a reform provision that is not included in the Build interview? Build by hand one or more except statements in the Run Spec editor. Be sure that the reform entry starts with a major comment. Consult the documentation of input parameters to identify the table field(s) that need to be used in the except statement(s). If you don't know what database table contains the policy parameter you need to change, use the "Search documentation" or "View documentation" items on the Help menu.

If you think that the reform provision missing from the Build interview should be included, notify Martin Holmer via email (holmer@polsim.com).

How do I tell what Reform Provision <2> means when the "Specify parameters of reform provision" dialog box appears after pressing the Add button? The <2> refers back to the <2> parameter listed for that provision in the Reform Provisions window. A parameter that is a table field can be clicked to view the documentation for that field.

How do I generate a cross-section sample containing individuals of all ages? Cross-section samples can be produced by either PENSIM or GEMINI when using the OLC mode of model operation. In a single OLC mode run, PENSIM (GEMINI) can produce up to five (three) different cross-section samples where the samples differ with respect to the survey year and/or the focus of the supplementary survey questions. The number, timing, and focus of the cross-section surveys is controlled by PENSIM input parameters OUTPUT.xs? (where ? ranges from a to e) and, if OUTPUT.xs? is true, by associated input parameters in the XS? table. The analogous GEMINI parameters are all in the STATS input database table. View the documentation of these tables for the details.

Here is a run specification file fragment from an OLC mode run that illustrates how to generate one PENSIM cross-section survey:

```
...
create pensim run same as run 59 except:
table RUN field ssasim_rid is @
table RUN field scenarios is 1 ## same as SSASIM:RUN.scenarios
# Cross-section survey revisions:
table OUTPUT field xsa is F
table OUTPUT field xsb is T
table XSB field supplement is 5 # asks about prior year saver's credit
table XSB field olc_mode is T # include individuals of all ages in sample
table XSB field olc_year is 2008 # conduct survey at start of 2008 so that
table XSB field olc_moment is 0.01 # prior year saver's credits are for 2007
table OUTPUT field xsc is F
...
```

How do I produce both solvency and distributional results for the same reform proposal when using microsimulation methods? The answer depends on whether you want to use CBO or OCACT methods to produce solvency estimates.

If you want to use standard microsimulation (i.e., CBO) methods to produce solvency results and need distributional results for only one birth cohort, you can produce both solvency and distribution results in the same run by choosing the RCS+OLC option in the Build interview preamble.

If you want to emulate OCACT actuarial methods when producing solvency results, then you need to specify separate runs: an OLC run that emulates OCACT actuarial methods and an RCS run using the same reform proposal. Do this by building the OLC run (including all the reform provisions) and then using it. Then use the method described in the previous section to construct a run specification file with the same reform provisions where SSASIM operates in the RCS mode. If you are specifying stochastic runs, be sure that the value of `num_years` in the SSASIM RUN table is the same in the two run specification files, otherwise the random number streams will be different in the two runs beginning in the second scenario. Always set the value of `num_years` equal to the larger of the two values that are specified during the Build interviews.

How do I change the ultimate value of one of the fifteen key assumption variables? Build by hand an assumption statement for that variable. For example, if you want to use all the assumptions in the base run except for the ultimate value of the inflation rate, then include the following assumption statement to change the ultimate value of the inflation rate to four percent:

```
assumption trnd_mv inf mean is 4.0
```

The assumption statement must have `assumption` and `trnd_mv` as the first two keywords. The third keyword can be any one of the fifteen three-letter assumption variable names described in the RSF language syntax section on page 16 and in the input database documentation for the SSASIM TRND_MV table. And the fourth keyword can be either `tran_years`, `mean`, or `std_dev`, corresponding to those three fields in the TRND_MV table. A “kink” in an assumption variable’s time path can also be specified using `except` statements for the appropriate table (for example, the PRDMRKT table for the inflation rate).

How do I specify a run with stochastic equity returns? Use the Build interview to specify a deterministic run (that is, a single-scenario run) that is what you want in all respects except for the stochastic equity returns. Then, in the RSF editor, add the following lines under the `create ssasim run` statement:

```
...
# stochastic equity returns:
assumption trnd_mv eqr mean is 0.0897 # rounded from 0.089651 (see below)
    # (1.029+0.035)*1.028=1.093792 ==> log value is 0.089651
    # above assumption of a 3.5% real equity premium is
    # the standard assumption made by both OCACT and CBO
assumption trnd_mv eqr dev_stddev is 0.1802 # for log(1+decimal_ror), which
    # is same as CBO equity volatility assumption of 20.2% for percent_ror
...
```

The 3.5 percent real equity premium is the standard assumption used by both OCACT and CBO, and is based on historical equity returns. However, some

finance economists believe that the equity premium, looking forward, is lower than 3.5 percent.

Use the `resce` utility from the Windows command prompt to see the geometric mean, arithmetic mean, and volatility of equity rates of return implied by other values of these two input parameters.

You also need to increase the number of scenarios from 1 to something like 500 or 1000 depending on the mode of operation. If you are specifying a GEMINI run (that is, an RCS mode run), be sure to pay attention to the `PENSIM:RUN.sample_pct` value because this is the sample size in **each scenario**. So, for example, 500 scenarios and a sample percent of 0.01 produces a total cohort sample of five percent, which is roughly a quarter of a million individuals for most birth cohorts. When specifying an OLC mode run, you **must** use the 0.1 sample size for reliable results; the consequences are lengthy run execution times even if you have a computer with two dual-core chips and have set `OLC.threads` to 4.

In all cases, it is best to start by experimenting with a run that has two or three scenarios. This will give you a sense of the run execution time on your computer and the size of the output files.

How do I specify exemplary cohort individual (ECI) attributes? In the RSF Toolkit Build interview, pick the ECI mode, look at the individual statements that are automatically generated, and then modify them to meet your needs.

How do I characterize reforms in CBA mode? When the PSG models are operating in the CBA mode (as is always the case in the free educational version), there is a limited range of input database parameters that affect social security benefits and payroll taxes. This limitation is rooted in the fact that the CBA mode, unlike the other three modes, does not use the policy parameters under the COHORT POLICY tables to calculate benefits for individuals with full life histories. In CBA mode, the average level of initial DI and OASI benefits can be adjusted using the `BEND_DI.initben_sf` and `BEND_RI.initben_sf` parameters; the COLA rules can be adjusted using the `BEND_DI.col_a_offst` and `BEND_RI.col_a_offst` parameters. In addition, the `BEND_DI.pay_pct` and `BEND_RI.pay_pct` parameters can be used to levy across-the-board benefit reductions (as might happen when the trust fund is exhausted and tax revenue is less than scheduled benefits). In CBA mode, payroll tax rates can be adjusted using the `TAXR_DI.pay_t_rate` and `TAXR_RI.pay_t_rate` parameters; raising the maximum taxable earnings level or imposing payroll taxes about the maximum are not practical to implement.

How do I view pension results in GEMINI output files? In the RSF Toolkit Build interview select the RCS mode of operation. If you are using the educational version of GEMINI, you must, after the Build interview is completed, change by hand the value of `table RUN field only_esp` from F to T.

How do I ...? ...

**PLEASE SUBMIT SUGGESTED “HOW DO I ...?” QUESTIONS
TO holmer@polsim.com**

RSF Language and Scope

This final section of this primer presents a more technical description of the run specification language and identifies the relatively small number of input database tables that cannot be named in RSF except statements.

RSF Format and Name. An RSF must be saved to disk as an ASCII text file and have a file name that ends with the .rsf extension. Do not copy text from a word processor and paste it into an RSF because word processors typically embed invisible non-ASCII characters in text.

RSF Language Syntax. The RSF language is not case sensitive and has the following formal syntax:

```
e-object := table <tname> field <fname>
a-object := assumption trnd_mv avarbl afield
avarbl := tfr|imm|mdr|fpr|mpr|unr|inf|pgr|wsg|hwg|nir|dif|drf|eqr|tbs
afield := tran_years|mean|std_dev|dev_stddev
svalue := fvalue|@
dvalue := [same thru <skeyvalue0> and] <fvalue1> for <skeyvalue1> [and ...]
fvalue := decimal_value|integer_value|lvalue
lvalue := true|t|false|f
modeln := ssasim|gemini|pensim
c-stat := create modeln run same as run <base_run_number> except:
e-stat := e-object is svalue|dvalue
a-stat := a-object is decimal_value|integer_value
i-stat := individual id fvalue notes ... age_kid4 fvalue
```

The `same thru` clause of `dvalue` can be omitted only when `<skeyvalue1>` is the minimum value of the secondary key specified by the base run and is not less than the minimum allowable value for the secondary key. When the `same thru` clause is used, the value of `<skeyvalue0>` must be no less than the minimum allowable value for the secondary key. An `svalue` of `@` is replaced by the new run number when the RSF is used to create a new run in the input database.

An RSF consists of one, two, or three model-specific sections. Each model section consists of one model-specific create statement (`c-stat`) and zero or more except statements (`e-stat`). Each table field can be named in at most one of these except statements. One or more assumption statements (`a-stat`) are allowed, but not required, following the SSASIM create statement. One or more individual statements (`i-stat`) are allowed, but not required, following the SSASIM create statement.

Blank lines in the RSF are legal. The `#` symbol denotes the beginning of a comment, which can start anywhere on a line and continues to the end of that line.

Limit on Table Scope in RSF Language. An RSF can refer to most of the database tables in the input databases used by the PSG models. But a number of input database tables have an irregular structure that prevents their contents from being manipulated using the RSF language. A list of the tables excluded from the scope of the RSF language appears below. And below that list is

a discussion of how the contents of those excluded tables can be manipulated using SQL statements embedded in a run specification file.

It is easier to list the tables *not* allowed in the RSF language statements because the total number of input database tables is large. All together, there are 93 SSASIM tables, 308 PENSIM tables, and 6 GEMINI tables.

1. **SSASIM** tables **not** allowed in RSF language statements:

- ASSUMP child tables: TRND_CC, DEV_P, DEV_CC
- POP child tables: POP_GA0, POPDYN, FR_A, IMM_GA, MDR_GA, SP_PROB, SP_AGED
- LABMRKT child tables: LABDYN, LFPR_GA, UNR_GA, EARN_GA
- BEN_DI child tables: NBEN_DI, ABEN_DI, BDYN_DI, RBEN_DI, INCR_DI, RMRT_DI, RECR_DI
- BEN_RI child tables: NBEN_RI, ABEN_RI, BDYN_RI, RBEN_RI, INCR_RI, DECR_RI
- TAXEARN
- FUND_DI child table: STB0_DI
- FUND_RI child table: STB0_RI
- FUND_RD child table: STB0_RD
- ACCT child tables: ACCTBAL, ACCTAPY
- COHORT/INDS/IND child table: EARNPCT
- HISTORY, HISTMR, HISTMDR

2. **PENSIM** tables **not** allowed in RSF language statements:

- INDBEHV table and **all its child tables except: EMIGRA, EMIGDYN, IMMIGRA, IMMIDYN**
- PENCHAR child tables: DCTYPE1, DCTYPE2, DBVEST1, DBVEST2, DBRETR1, DBRETR2, DBSIMM1, DBSIMM2, CACONT1, CACONT2, STECLT1, STECLT2

3. **GEMINI** tables **not** allowed in RSF language statements:

- none

No out-of-scope GEMINI tables means that any GEMINI run can be specified using the RSF language without resorting to embedded SQL statements.

Embedded SQL Statements. A run specification file can include SQL statements if they are placed in an SQL-BEGIN ... SQL-END block. Any number of SQL blocks can be used following the RSF create statement for SSASIM or following the RSF create statement for PENSIM, but no SQL blocks are allowed

following the GEMINI create statement because all GEMINI tables are in the scope of the RSF language.

Note that within an SQL block `@` is replaced by the new run number when the RSF is used to create a new run in the input database.

This embedded SQL extension to the RSF language will be required only in unusual circumstances. Probably just two situations will account for the vast majority of the few circumstances where embedded SQL statements are required to specify a run. Here we illustrate the use of embedded SQL statements in these two situations.

1. Specifying Stochastic Correlations between Assumption Ultimate Values.

- Suppose that in a stochastic CBA-mode run inflation and the nominal interest rate are the only two of the key assumption variables that are assumed to have stochastic ultimate values. This could be accomplished in SSASIM by setting `RUN.scenarios` to one thousand and by setting `TRND_MV.std_dev` to positive numbers for the `inf` and `nir` assumption variables. But typically there is a positive correlation between the rate of inflation and the nominal interest rate. How can a positive correlation coefficient between the `inf` and `nir` assumption variables be specified when the `TRND_CC` table is outside the scope of the RSF language?
- The following statements (placed somewhere below the SSASIM create statement) will generate positively correlated ultimate values for the inflation rate and the nominal interest rate.

```
create ssasim run same as run NN except:
# ...
assumption trnd_mv inf std_dev is X.XXXX
assumption trnd_mv nir mean is -Y.YYYY
assumption trnd_mv nir std_dev is Z.ZZZZ
# ... Add positive correlation +0.CC between the
#   inf (variable number 7 in trnd_cc table) and
#   nir (variable number 11 in trnd_cc table)
#   ultimate values.
SQL-BEGIN
INSERT OR REPLACE INTO trnd_cc
  VALUES( 1369, 11, 7, '', +0.CC );
-- note that the trnd_cc table documentation
-- requires that the 11 be before the 7
-- because trnd_cc is a lower-triangular matrix
SQL-END
table ASSUMP field trnd_cc_id is 1396
# ...
```

- Each of the `SQL-BEGIN` and `SQL-END` statements must be alone on its own line. They are shown here in all upper case, but that is not

necessary. Each SQL statement inside the SQL block must end with a semi-colon. SQL language keywords are traditionally all upper case (as shown here), but that is optional.

- Notice that the embedded SQL statements must precede any reference to the rows created by the embedded SQL statements.
- For more details on specifying runs where the assumption variables vary stochastically, see the `stoch10cba.rsf` file distributed with the PSG models.

2. Specifying Alternative ECI Lifetime Earnings Histories.

- Suppose that in an ECI-mode run all individuals but one are assumed to have the OCACT “scaled” earnings histories that are already stored in the EARNPCT table. How can the arbitrary earnings history of the other individual be specified when the EARNPCT table is outside the scope of the RSF language?
- The following statements (placed somewhere below the SSASIM create statement) will generate an individual with a “steady” earnings history.

```

create ssasim run same as run NN except:
# ...
SQL-BEGIN
/*
Add steady earnings history at 150 percent of the
average wage index beginning at age 21.
*/
INSERT OR REPLACE INTO earnpct VALUES(150, '',16, 0);
INSERT OR REPLACE INTO earnpct VALUES(150, '',20, 0);
INSERT OR REPLACE INTO earnpct VALUES(150, '',21,150);
SQL-END
# Note individual below has earnpct_id = 150,
# which is defined in the SQL block above.
individual id 203 notes ~id=203~
    male T educ 4 earnpct_id 150
    disab_age 999 retire_age 67 death_age 85
    trace_db F trace_dc F trace_scen 1
    spouse_id_ 0 age_kid1 99 age_kid2 99
    age_kid3 99 age_kid4 99
# ...

```

As the examples above should suggest, the embedded SQL capability requires knowledge of the structure of the database tables and familiarity with relational databases and structured query language (SQL). Each PSG model uses its own SQLite3 input database that enforces referential integrity. See the SQLite home

page <http://www.sqlite.org> for more details, especially on the SQLite dialect of SQL.