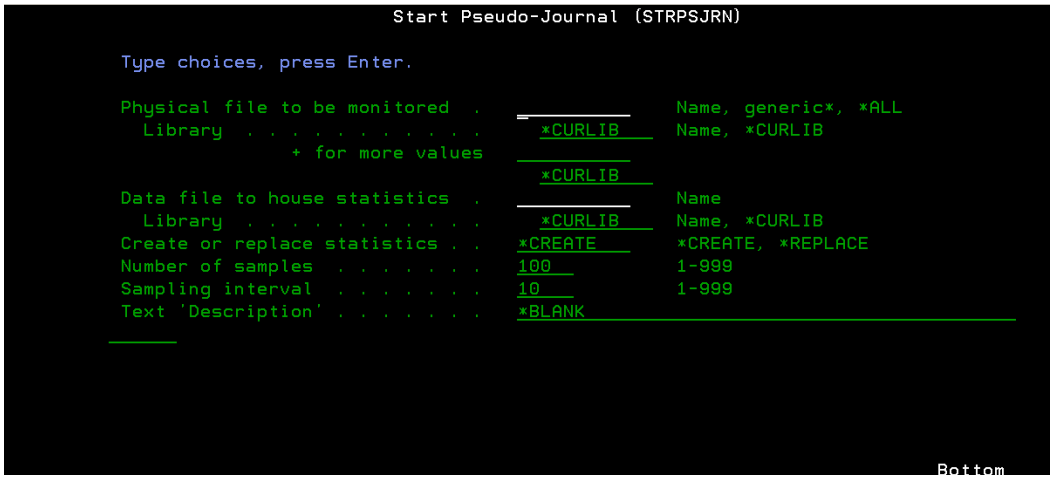
How to use the Pseudo Journal Tool

Now that you have installed the "Pseudo-Journal Tool" you are able to use the two commands associated with Pseudo journaling. The first will help you to collect the metadata we need. The second will help you display the results in a variety of ways. Both commands are found in the PSEUDOJRN library.

The command to collect the metadata we need is called Start Pseudo Journal and the matching CL command is **STRPSJRN.** This command can be long-running, so it's recommended that the command be executed in a batch process. Like any i5/OS CL command, you can prompt the command with the F4 key to view the command parameters. The prompted view is shown in the figure below.



On this screen you can see some blank fields that must be filled in by you, but don't worry, we'll explain them one by one.

First of all, notice that this tool is called "Pseudo Journal" because it helps you estimate what would happen on behalf of your data base tables if they were journaled.

Let's look at each parameter on the screen.

The first parameter, called "Physical file to be monitored", allows you to specify the name of the physical file(s) you want the tool to keep an eye on. These are the "candidate" physical files you're thinking of journaling. They're generally going to be the principle files which get modified when your application runs (because that's probably what you're contemplating having us journal some day).

If there are only a few files you have in mind, listing each of them isn't much of a chore. However, if you have hundreds of such files, identifying each individually

would be a burden. That's why the command provides three options that can be chosen to help simplify this process. They are are listed below

- Name (Specify the name of the physical files to monitor)
- *ALL (Signifies that all the physical files residing in the designated library should be monitored)
- Generic* (Specifies a generic name of the set of physical files to monitor - - helpful if the subset of files you have in mind all start with some common prefix).

Caution: Don't go overboard at metadata collection time. The more files you ask the tool to monitor, the more total work you're asking it to schedule. Use the Goldilocks principle: Not too much, not too little, but just right.

The next parameter is called "Library", here you need to specify the name of the library containing the physical files to be monitored. This will generally be your application library.

The third parameter called "Data file to house statistics", identifies the name of the file that will save the necessary metadata to create the reports and statistics in a later step via the second command. That is, this is the place where the collected metadata will reside between the first (collection) step of the tool and the final (display) step of the tool.

The fourth parameter, also called "Library", specifies the library that will house the file in which the metadata and statistics reside. That is, this is the library for the physical file identified in the third parameter.

The fifth parameter lets you designate whether you want the system to create or replace the metadata data file. If you select *CREATE a new metadata file will be created. If you specify *REPLACE the existing metadata file will be overlaid.

The sixth parameter: "Number of samples", is where you indicate the quantity of times you want to tool to wake up and capture another data point. The idea here is that over the course of executing your application you may want the tool to take a series of successive measurements at fixed intervals (say, 20 samples, at 5 second intervals for a total duration of 100 seconds). This sixth parameter allows you to designate the quantity of samples. You may specify a value between 1 and 999.

The seventh parameter, "Sampling interval", allows you to designate the time gap between samples. It must be a value, expressed in seconds, between 1 and 999. If you needed fine-grained delta values you could have the collection tool wake up every second. If you needed only general trends you might want to advise the tool to wake up only every minute.

Caution: If you ask the tool to monitor lots of tables, it would be impractical to expect it to wake up every second and ripple through 10,000 files. Hence you need to strike a practical balance between frequency of data collection and quantity of files being monitored. If you're monitoring only a few files, you can employ a small sampling interval (a few seconds). On the other hand, of you've asked the tool to monitor thousands of files, a sampling interval of 15-30 seconds (or longer) may be more practical.

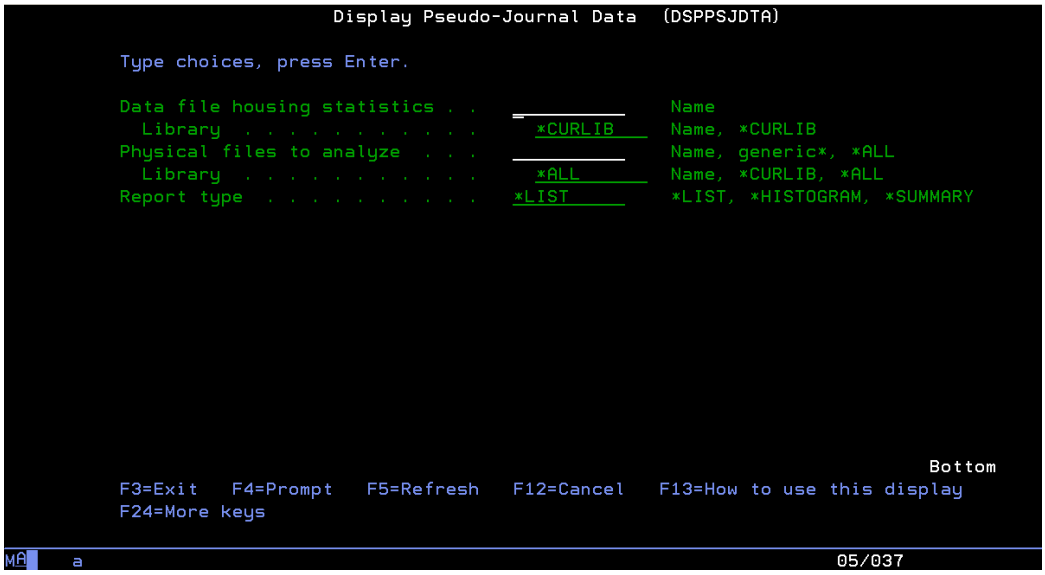
There's only one parameter on this first command left to discuss, and it's called "Text description", here you have the option to write a short description about the kind of application you are monitoring. Use of this field will help you recall later why you took the sample and what it represents. For example, you might want to include a description such as 'Company xyz – 20 minute run during payroll processing on 8/3/07 at 2pm'.

Now you are ready to use the tool….

Note:
** This CL command (the "Start" Pseudo Journal) only collects the raw data needed to create the summary statistics. There's a matching CL command (described below) called "DSPPSJDTA" (Display Pseudo Journal Data) that allows you to reduce the statistics to summaries and graphs of the observed Journal estimated overhead.*

How to display the results

DSPPSJDTA is the CL command that will let you see the summarized results derived from the raw statistics collected by the Start Pseudo Journal command. After you press F4 you will see a screen as shown below.



Now it's time to explain each parameter for this command.

You'll need to remember the name you granted to the metadata file which was produced with the Start Pseudo Journal command, hence we are going to enter the name of this metadata file in the field labeled "Data file housing statistics".

As you can see the second parameter refers to the library that contains the metadata file specified above and it's called "Library".

The third parameter, called "physical files to analyze", lets you choose the specific physical files that you are interested in analyzing. This is, even though you instructed the first command to keep an eye on 100 files, you may want to zero in on only a few of them at analysis time. If that's your pleasure, you can customize the analysis to keep from cluttering your screen with data from all 100 files you monitored. Here too, there's an *ALL option if you simply want to see results for all of the files that were monitored.

Let's say, for example that when you started Pseudo Journaling you elected to monitor 100 files. In the meantime you've concluded that many of the 100 files monitored had little if any modifications taking place and hence very little journal traffic. As a consequence, it's quite possible that you now want to concentrate on only two of those 100. It's this third parameter that allows you to customize your

view so that your subsequent screens and reports will only focus on the two of interest and not be cluttered with data from all 100.

As you did before, you have to specify the name of the library that contains one or more of the physical files whose summarized statistics you now want to see. You can, of course simply choose *ALL. This parameter is called "library".

The next parameter is called "Report type". This is one of the most important parameters because here you can choose one of the three ways to see the results based upon the data collected by the **STRPSJRN** command.

- The first way to see the results is a **Summary (*SUMMARY)** that contains all statistics. This choice does not break out the statistics table-by-table but rather rolls up one big summary for the whole application. It's a good way to get a high-level initial view of what the application has done and how much total journal traffic it's likely to generate. It even attempts to estimate the quantity of additional CPU overhead which would ensue as a direct result of having journal enabled.

- The second one is a **List (*LIST)** choice that shows, table-by-table, the statistics. That is, if you feel you need to know which table produced the most journal traffic and which produced the least, seeing the traffic broken out table by table might be helpful. The *LIST option produces such a list -- a list of tables which were monitored and their matching estimated journal contributions.

- And finally you can elect to see a **Histogram (*HISTOGRAM)** that shows the results graphically. This is probably the choice you would make if you had taken hundreds of samples and want to visually see where there had been peaks and valleys of journal traffic or whether the application tended to manifest rather steady-state behavior from a journal perspective.

Once you select one of these three reporting types, you'll be presented with additional customization parameters unique to that style of reporting.

For example, the first subsequent parameter that is shown when you select the **Summary** report type is the one called "Output" that lets you choose between the standard output (to a screen) or a spooled file whose contents could be printed.

Had you selected the ***LIST** report type you'd see a refinement parameter known as "Sort by". It provides an opportunity to influence the ordering of the resulting list of tables being monitored. For example, you might simply want to list them in alphabetical order by table name, or perhaps you want them sorted by quantity of journal activity they experienced during the run so that the strongest journal contributor stands out as the first table in the list. That's why the two options

offered are: to sort your data by workload (*WORKLOAD) or to sort your tables alphabetically (*ALPHA).

If you elected to see the results displayed as a histogram you'd also get some customization options. For example, you'll see a parameter allowing you to customize the histogram type that you want displayed. There are three choices.:

- *DATA (Shows the statistics about the estimated quantity of bytes deposited into the journal receiver and hence the quantity of extra bytes written to disk if journaling was enabled.)
- *DSKW (Shows the statistics about quantity of separate journal writes requests to disk - - which may be a good way to sense how much busier you disk arms are likely to become.)
- *DSKWCACHE (Shows the statistics about quantity of journal writes to disk which would have been scheduled if the performance enhancing option known as "journal caching" had been enabled)

We trust that this planning tool will help you make "what-if" decisions regarding the files you're thinking about journaling.