

Getting Started with Xi-Batch – A Brief Introduction

This guide will help you get started with Xi-Batch by guiding you through some easy examples of basic scheduling and job monitoring commands.

It's assumed that you've completed the installation, and that the job scheduler is running.

Creating a Variable

Let's start by creating a scheduling variable. This is a value stored by Xi-Batch that can be used to control the flow of jobs in the system. Later, we'll use this variable to demonstrate how job control is accomplished.

Make sure your PATH includes a path to the Xi-Batch programs. The default location for these is /usr/local/bin but you had the option at installation time to pick another location. It will be easier to see what's going on if you have access to two terminal windows on your system. We'll run the job monitoring program 'btq' in one window, and issue commands at the shell prompt in the other window. If you have X-Windows running, it will be nicer to look at if you run the graphical monitor 'xmbtq' instead of the character-mode 'btq'. Set this window to variable view (rather than job view) by typing an uppercase "V" command.

To create a variable named TEST1 and assign it an initial value of zero with a descriptive comment, enter this command at the prompt:

```
btvar -C -c "Used for initial testing" -s 0 TEST1
```

The uppercase -C parameter tells the command to create a new variable. The lowercase "-c" parameter specifies the comment. The -s argument specifies the value that we are storing in our variable, and the final argument is simply the name we wish to use.

In the terminal window running btq, you can see the variable just created. (You may need to scroll the display.)

The command:

```
btvar TEST1
```

will return the current value of the variable TEST1.

Let's change the value in TEST1 before we move on to jobs. Use the command

```
btvar -s 1 TEST1
```

to set the variable to a value of 1, then run

```
btvar TEST1
```

to verify that the variable has been changed. (If you've been watching the other window, you can see the variable has been updated immediately.)

Creating a Job

Before we create a job in Xi-Batch, you should switch your view in the btq window to jobs view by entering an uppercase “J” in that window. (If you’re using the graphical window, jobs are displayed in the upper part of the screen by default.)

Enter this command:

```
btr -h firstjob
```

Since we have not specified the name of a script to run, the btr command is expecting input from the terminal. You will be prompted, “Warning: expecting input from terminal” to remind you. (The `-h` argument specifies a title for our job, which will be displayed in any job display that we run.)

Let’s enter

```
sleep 120
```

and then “control-d” to terminate the input.

In the job display window, you will see (for the next 120 seconds, anyway) an entry in the display showing the job we’ve just created. It will be displayed with the default priority and load level of the user that submitted it.

Now, let’s create a script to do the same thing we just did on the command line. Using your favorite text editor, create a job called “sleep120” and insert the line

```
sleep 120
```

into the file. Close and save the file. Now, enter the command:

```
btr -h scriptjob sleep120
```

As you can see from the job display, we have just created a job and given it a script to run. Also note that the first job we created has completed and been deleted from the system. This is because we neither told the system to retain the job, nor specified a repeat interval for the job to run again. It’s also the default for the system to run a job immediately if you haven’t specified a specific run time for it.

Setting a Run Time

Enter the command

```
btr -h timedjob -T 12:34 sleep120
```

replacing the time “12:34” with a time value about five minutes from now.

When you enter this command, you’ll see in the job display that your job is entered into the system, and under the “Time” column, you’ll see the start time that the job will be made eligible to run. (We say “made eligible to run” instead of “started” because there may be other system constraints, variable conditions, etc. in effect that may prevent the job from running at the exact time specified.) In this learning exercise, though, we can be confident that the job will start at the appointed time, and you can observe this when the specified time arrives.

Changing a Variable

Now, we will demonstrate the beginnings of how the Xi-Batch system controls job flow using variable checking and assignment. Let's set our test variable to the value of zero again with the command

```
btvar -s 0 TEST1
```

Now, let's enter our job again, and this time, we'll tell the system to change the value of TEST1 when the job exits normally.

Use the command

```
btr -h vartest1 -f N -s 'TEST1=1' sleep120
```

In this example, the argument "-f N" indicates that the following "set" arguments are to be performed at a Normal job completion. The "-s 'TEST1=1'" argument specifies the variable to be set, and the value to assign to it.

When you run this command, you'll see that the value of TEST1 gets set to a value of 1, exactly as if you had run the 'btvar' command explicitly.

Simple Job Chaining

Now, let's put together the concepts of variable assignment and variable testing to make one job wait on another.

First, we'll submit a job that will wait on the queue until the variable TEST1 takes on a value of "3".

```
btr -h waitjob -c 'TEST1=3' sleep120
```

In the job display, you'll see TEST1 listed under the "Cond" column, showing that the job is waiting on a variable test condition to be satisfied.

Now, we could either use the 'btvar' command to change the value of TEST1 to 3 (you might include such a command within a script in a production system), but let's run a job and let the system set the variable upon normal job exit, just as we did in the previous example.

```
btr -h chaintest -f N -s 'TEST1=3' sleep120
```

This job will start immediately (since it is not conditioned on any variables, time specifications or system constraints). When it exits in 120 seconds, the system will set the variable TEST1 to a value of 3, indicating that the previous job we submitted with the title 'waitjob' is now eligible to run.

Now, go explore and have fun!

This small set of examples was designed to get you familiar with the concepts of jobs, time specifications, scheduling variables, and the use of the job monitoring screens. here are some other aspects of Xi-Batch that you will want to explore. All of them are

described in detail in the System Reference Manual, included in HTML format on your distribution media.

Here are some important features to read about and try:

- Security – Jobs and variables are owned by the user that creates them, and by default are invisible to other users on the system, except for their names.
- Multi-hosting – We ran our examples on a single system, but in a networked environment, jobs and variables can exist on any host. A job submitted on host A, for example, can be held waiting on a variable defined on host B.
- Job repetition. When a job is submitted, you can specify a repeat interval, in minutes, hours, days, weeks or months. Xi-Batch automatically handles holidays (if you've defined them) and weekends.
- The graphical program 'xmbtr' is a convenient tool for submitting jobs and setting the time specifications and variable testing and setting functions associated with the job. It is easier than remembering all of the command-line arguments.
- The program 'btvar' can be used within a job script to set variables to specific values. Likewise, the command 'btr' can also be included in a job script to cause the submission of other jobs.
- By default, job output (both stdout and stderr) is e-mailed to the user who submitted the job. Try this by submitting a job that does a 'ls' command to list a directory's contents.
- Xi-Batch can establish a custom environment for each job to run under.
- If Xi-Batch is shut down, jobs that are running are terminated. Jobs that are in the queue remain there, and can run the next time the system is started. See the documentation for tips on how to conduct an orderly shut-down of Xi-Batch.
- Load leveling and job priorities are designed to provide tools to manage resource utilization of your systems. Properly used, they can allow users to submit jobs at will, and give your operations staff the tools to control how many jobs are run at a given time.

Please call Taricon Technologies if we can assist you in any way during your evaluation. Thank you for asking to evaluate our products.