**Shop Runner Product Feed Write-up**

**1. Environment issues:**

a) Classpath configuration.

The following line is missing from production job start-up scripts. The consequence is that the jar file must include the XSQLConfig.xml file in order for the program to function properly. To resolve this issue in PROD, this line needs to be added to PROD start scripts:

CLASSPATH=$CLASSPATH":"/opt/jobs/mbp/datafeeds/resources/xsql

The additional allows the hour\_glass scheduler to locate the proper config file(s) located in that folder.

b) Invalid folders.

In the DEV and PROD environments there are unnecessary/confusing folders:

hour\_glass/resources

c) Please also note the important detail that the DEV hour\_glass environment's path contains a lower case "mbp" while the PROD environment contains an upper case "MBP". This is an issue since unix/linux is case sensitive and the paths specified in the config files etc. are not going to work in PROD unless this details is first resolved.

**2. Important Details:**

**2.1 Process Label:**

Job scripts main java call has following syntax:

java -D[LABEL\_JOB\_NAME\_HERE] [memory settings] -classpath "$CLASSPATH" hourglass.binary [config.xml.file]

java -DOUTBOUND\_DATAFEED\_HOURGLASS -Xmx1024M -classpath "$CLASSPATH" com.flowers.schedule.HourGlass data/outbound\_datafeed\_hourglass.xml

The job label is extremely important to recognize because the user would be unable to identify which specific process to terminate with the presence of the label.

The memory setting is also important because the feed process consumes more than the default memory allocation that the JVM allocates to the process.

ps -ef displays the processes in memory.

grep [LABEL\_JOB\_NAME\_HERE] searches for the [LABEL\_JOB\_NAME\_HERE] in a predetermined text.

"|" pipe command does a pseudo connect between two processes.

ps -ef | grep [LABEL\_JOB\_NAME\_HERE] therefore searches for the user label in the list of proceses making it easy to terminate the required process (kill -9 [processid])

**2.2 Important Changes:**

The Shop Runner product data feed is different from previous feed functionality in the following ways:

a) It does not need to be called multiple times for each brand. It only needs to be supplied with the brand code comma separated list and be called once.

b) It can be configured so that each feed file output can be customized to specify the sftp host value and credentials.

The benefit of this approach is that the very resource expensive database access will be called once for the entire process and not several times as the previous approach required. There was also a requirement from shop runner for each brand to sftp the feed file to various host servers.

**2.3 Feed Types and Implementation:**

All the existing outgoing feeds use the ProductDataFeed.java feed handler except the following:

ShopRunnerDataProductDataFeed

RichRelavenceDataFeed

MercentDataFeed

These feeds utilize the MBPProductDataFeed java class.

**2.4 Generating XMLBeans:**

In order to generate the XMLBean classes, please follow the following instructions::

1. unzip xmlbeans-2.0.0.zip. This will be your XMLBEANS Home in the script that generated the xml beans jar

2. Create a folder for the project that needs XML Beans.

a. Place a copy of the Generate-XMLBeans.bat. This will be customized for each project and a specific \*.xsdconfig file will need to be configured. This has configuration parameters for the XML beans. We use it to control the package names among other things used in the generated classes.

b. I also have a sub folder for the XML schema files(s).

c. Modify the environment variables in the Generate-XMLBeans.bat file for your system. There are comments in the bat file to help you with that.

d. run the Generate-XMLBeans.bat from the command line to generate the jar file.

**2.5 Miscellaneous:**

1) A custom feed call has the following syntax:

nohup ./run\_sample\_datafeed\_job.sh >logs/sample\_datafeed.log 2>logs/sample\_datafeed.err &

Calling "nohup [command/script]" starts the command or script as a daemon process that executes in the background and returns the control prompt to the user. This process will continue to execute until the user session has ended. It is therefore important to note that the presence of the "&" at the end of the command instruct the unix/linux operating system to keep the process executing after the user session has ended. The presence of the ">" instructs unix/linux to redirect the process' output from standard output (screen) to a file.