#!/tools/perl/bin/perl

#version 1.0

#

#

###########################################################################

#

# State Street Bank and Trust Company

#

###########################################################################

#

# Filename: job\_control.pl

#

# SCCS information:

#

# version: 1.44

# date changed: 03/02/05 13:56:57

# date retrieved: 03/02/05

#

#

# Description:

#

# This Perl script is used to launch, monitor and kill processes.

#

###########################################################################

#

# R E V I S I O N L O G

#

# Date Name Description

# -------- --------------- -------------------------------

# 09/16/14 Nick Iagallo Modified for Cloakware.

# 11/20/09 Nick Iagallo Added $process\_path\_nm so that processes can be

# run from different directories

# 02/02/05 Nick Iagallo Added $Path when using 'nak'.

# 02/02/05 Nick Iagallo Removed else statement from 'path setup' section.

# 06/04/04 Nick Iagallo Added an input parameter to the warehouse

# manager daily process

# 04/16/04 Nick Iagallo Added an input parameter to a daily process

# go\_pe\_generic and go\_pe\_client\_generic.

# 04/02/04 Nick Iagallo Added an input parameter to a daily process

# 03/25/04 Nick Iagallo Added code to start the daily script

# go\_pe\_client\_generic.

# 09/24/03 Nick Iagallo Added an & when starting a workbook go\_script

# 09/18/03 Nick Iagallo Added the -w option to grep in monitor\_jobs function

# 09/17/03 Nick Iagallo Corrected ps statement in monitor\_jobs function.

# 09/11/03 Nick Iagallo Fixed the killing and monitoring of java

# processes.

# 07/24/03 Nick Iagallo Changed perl version being used

# 07/03/03 Nick Iagallo Added code to the monitor\_jobs function

# 07/03/03 Nick Iagallo Corrected script name(go\_workbook\_generic.sh)

# 06/26/03 Nick Iagallo Modified the monitoring and killing of

# processes

# 06/23/03 Nick Iagallo Added code to start the daily script

# go\_workbook\_generic.csh

# 06/16/03 Nick Iagallo Removed the code to not kill a job if it is

# currently processing a request.

# 05/19/03 Nick Iagallo Added code to not kill a job if it is currently

# processing a request.

# 03/25/03 Nick Iagallo Moved the OSR process under job type 'R'

# 03/25/03 Nick Iagallo Added the ability to launch the OSR process

# 03/20/03 Ari Silverman Removed ccms\_path to everything besides go\_pe\_generic

# 03/20/03 Ari Silverman Added ccms\_path to everything besides go\_pe\_generic

# 03/18/03 Ari Silverman Added ARGV[3] Path

# 03/14/03 Ari Silverman Added another Parm for FU,LU and SP

# 11/08/02 Nick Iagallo created

# 11/18/02 Nick Iagallo Added warehouse manager launch

# 11/21/02 Nick Iagallo Added kill and cleanup processes

# 11/21/02 Nick Iagallo Corrected kill bug

# 11/25/02 Nick Iagallo Removed an input parameter when calling

# warehouse manager go\_script

# 11/27/02 Nick Iagallo Added an input parameter when calling

# cleanup script

# 12/12/02 Nick Iagallo Added a ccms return address to mailx

# 12/17/02 Nick Iagallo In monitor\_jobs function: tightened grep

# 01/21/03 Nick Iagallo Changed if statement to look for pattern

# "go\_pe\_generic"

# 01/27/03 Nick Iagallo Added new type of daily process.

# 02/18/03 Nick Iagallo Put daily process in background when

# process is not a go\_script

###########################################################################

# Job Types: D - Launch daily jobs

# M - Monitor jobs

# K - Kill daily jobs (NAK)

# R - Launch reports

# C - Launch cleanup script

#

use Sybase::DBlib;

#---------------

# Read arguments

#---------------

$central\_server = @ARGV[0];

$job\_type = @ARGV[1];

$debug\_level = @ARGV[2];

$Path = @ARGV[3];

open(OUTFILE3,">$Path/echo\_log.log");

#-----------

# Setup path

#-----------

print OUTFILE3 "Path<$Path>\n";

if($Path eq "") {

if (((getpwuid($<))[0]) eq 'ccms')

{

$Path = "/usr/local/ccms/exe";

}

else

{

$Path = ".";

}

}

print OUTFILE3 "Path 2<$Path>\n";

#----------------------------

# Setup central server handle

#----------------------------

$userC = (getpwuid($<))[0];

if($userC eq 'ccms')

{

$exe\_path = "/usr/local/ccms/security/exe";

$pass\_file = "/usr/local/ccms/security/dat/.PASSWORD\_MATRIX";

$lib\_path = "/usr/local/ccms/lib";

}

else

{

$exe\_path = "/ssb/cm/security/password/exe";

$pass\_file = ".PASSWORD\_MATRIX";

}

$userC = "ccms";

$ENV{"PW\_MATRIX"}="$pass\_file";

$ENV{"LD\_LIBRARY\_PATH"}="$lib\_path";

print OUTFILE3 "PW\_MATRIX=$pass\_file\n";

print OUTFILE3 "LD\_LIBRARY\_PATH=$lib\_path\n";

print OUTFILE3 "$central\_server $userC\n";

$passwdC = `${exe\_path}/pwEcho.exe $central\_server $userC`;

$rc = $? ;

print OUTFILE3 "$passwdC\n";

if ($passwdC =~/^NA$/)

{

print OUTFILE3 "This password is not available\n";

exit -99;

}

close (OUTFILE3);

$sqlC = new Sybase::DBlib $userC, $passwdC, $central\_server;

if ($job\_type eq "D")

{

# Retrieve and launch all daily jobs.

#------------------------------------

$query = "SELECT server\_nm, ccms\_path\_nm, log\_path\_nm, process\_nm,

task\_typ\_cd1, task\_typ\_cd2, app\_db\_nm, process\_parm1,process\_parm2,

process\_parm3,process\_parm4, process\_path\_nm

FROM pe\_admin..job\_monitor

WHERE job\_typ = 'D'";

#print STDOUT "$query\n";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

$row = 0;

while(@dataC = $sqlC->dbnextrow(1))

{

$server[$row] = @dataC[1];

$ccms\_path[$row] = @dataC[3];

$log\_path[$row] = @dataC[5];

$process\_nm[$row] = @dataC[7];

$task\_type\_cd1[$row] = @dataC[9];

$task\_type\_cd2[$row] = @dataC[11];

$app\_db[$row] = @dataC[13];

$param1[$row] = @dataC[15];

$param2[$row] = @dataC[17];

$param3[$row] = @dataC[19];

$param4[$row] = @dataC[21];

$process\_path\_nm[$row] = @dataC[23];

$row++;

}

# Create log file

open(OUTFILE2,">$Path/job\_launch.log");

for ($x = 0;$x < $row;$x++)

{

if ($process\_nm[$x] =~ /^go\_pe\_generic/)

{

$cmd = "$process\_path\_nm[$x]/$process\_nm[$x] $ccms\_path[$x] $log\_path[$x] $server[$x] $task\_type\_cd1[$x] $task\_type\_cd2[$x] $app\_db[$x] $param1[$x] $param2[$x] $param3[$x] $param4[$x]";

}

elsif ($process\_nm[$x] =~ /^go\_pe\_client\_generic/)

{

$cmd = "$process\_path\_nm[$x]/$process\_nm[$x] $ccms\_path[$x] $log\_path[$x] $server[$x] $task\_type\_cd1[$x] $task\_type\_cd2[$x] $app\_db[$x] $param1[$x] $param2[$x] $param3[$x]";

}

elsif ($process\_nm[$x] =~ /^go\_pe\_whm\_generic/)

{

$cmd = "$process\_path\_nm[$x]/$process\_nm[$x] $ccms\_path[$x] $server[$x] $param1[$x]";

}

else

{

$cmd = "$process\_path\_nm[$x]/$process\_nm[$x] $param1[$x] $param2[$x] $param3[$x] $param4[$x] &";

}

system($cmd);

$runtime = localtime(time);

print OUTFILE2 "Command:$runtime==>$cmd\n";

}

close (OUTFILE2);

}

elsif ($job\_type eq "M")

{

$outfile = "$Path/job\_monitor.rpt";

open(OUTFILE,">$outfile");

$ret = &monitor\_jobs;

#---------------------------------------

# email report to production support if

# problems are found and generate alert.

#---------------------------------------

if ($ret)

{

close(OUTFILE);

&email;

exit -99;

}

close(OUTFILE);

}

elsif ($job\_type eq "K")

{

open(OUTFILE3,">$Path/job\_kill.log");

&Running\_Processes;

#-------------------------------------------

# Retrieve processes that need to be killed.

#-------------------------------------------

$query = "SELECT server\_nm, process\_nm, task\_typ\_cd1,

task\_typ\_cd2, process\_parm1

FROM pe\_admin..job\_monitor

WHERE job\_typ = 'K'";

#print STDOUT "$query\n";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

$row = 0;

while(@dataC = $sqlC->dbnextrow(1))

{

$server[$row] = @dataC[1];

$process\_nm[$row] = @dataC[3];

$task\_type\_cd1[$row] = @dataC[5];

$task\_type\_cd2[$row] = @dataC[7];

$param1[$row] = @dataC[9];

$row++;

}

for ($x = 0;$x < $row;$x++)

{

if ($param1[$x] =~ /pidfile/)

{

open(INFILE,"<$param1[$x]");

$line = <INFILE>;

close(INFILE);

$cmd = "kill $line";

system($cmd);

}

elsif (!($process\_nm[$x] eq "pe\_whm\_srv.exe"))

{

$cmd = "$Path/nak -g $process\_nm[$x] $server[$x] $task\_type\_cd1[$x] $param1[$x] -s 1 9 -a t -t 1 -f s";

system($cmd);

}

else

{

$cmd = "$Path/nak -g $process\_nm[$x] $server[$x] -s 15 9 -a t -t 1 -f s";

system($cmd);

}

$runtime = localtime(time);

print OUTFILE3 "$runtime==>Killed process:$process\_nm[$x] task type:$task\_type\_cd1[$x] $param1[$x] Server:$server[$x]\n";

}

#------------------------------------------------

# Reset calcs/reports back to 'Ready to Process'.

#------------------------------------------------

$query = "UPDATE pe\_report..sched\_queue

SET queue\_status\_code = 'R'

WHERE queue\_status\_code in ('A', 'W')";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

close (OUTFILE3);

}

elsif ($job\_type eq "C")

{

$cmd = "$Path/CleanLogFiles.pl $central\_server $Path/";

system($cmd);

}

elsif ($job\_type eq "R")

{

#------------------------------------

# Retrieve and launch all report jobs.

#------------------------------------

$query = "SELECT server\_nm, ccms\_path\_nm, log\_path\_nm, process\_nm,

task\_typ\_cd1, task\_typ\_cd2, app\_db\_nm, process\_parm1,process\_parm2,

process\_parm3,process\_parm4

FROM pe\_admin..job\_monitor

WHERE job\_typ = 'R'";

#print STDOUT "$query\n";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

$row = 0;

while(@dataC = $sqlC->dbnextrow(1))

{

$server[$row] = @dataC[1];

$ccms\_path[$row] = @dataC[3];

$log\_path[$row] = @dataC[5];

$process\_nm[$row] = @dataC[7];

$task\_type\_cd1[$row] = @dataC[9];

$task\_type\_cd2[$row] = @dataC[11];

$app\_db[$row] = @dataC[13];

$param1[$row] = @dataC[15];

$param2[$row] = @dataC[17];

$param3[$row] = @dataC[19];

$param4[$row] = @dataC[21];

$row++;

}

# Create log file

open(OUTFILE2,">$Path/job\_report.log");

for ($x = 0;$x < $row;$x++)

{

if ($process\_nm[$x] =~ /^go\_pe\_osr/)

{

$cmd = "$Path/$process\_nm[$x] $ccms\_path[$x] $log\_path[$x] $server[$x] $param2[$x] $param3[$x] $param1[$x]";

}

else

{

$cmd = "$Path[$x]/$process\_nm[$x] $server[$x] &";

}

system($cmd);

$runtime = localtime(time);

print OUTFILE2 "Command:$runtime==>$cmd\n";

}

close (OUTFILE2);

}

exit 0;

sub monitor\_jobs

{

#----------------------------------------------------

# Retrieve all jobs that should currently be running.

#----------------------------------------------------

$query = "SELECT server\_nm, ccms\_path\_nm, log\_path\_nm, process\_nm,

task\_typ\_cd1, task\_typ\_cd2, app\_db\_nm, process\_parm1,

process\_parm2,process\_parm3,process\_parm4

FROM pe\_admin..job\_monitor

WHERE job\_typ = 'K'";

#print STDOUT "$query\n";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

$row = 0;

while(@dataC = $sqlC->dbnextrow(1))

{

$server[$row] = @dataC[1];

$ccms\_path[$row] = @dataC[3];

$log\_path[$row] = @dataC[5];

$process\_nm[$row] = @dataC[7];

$task\_type\_cd1[$row] = @dataC[9];

$task\_type\_cd2[$row] = @dataC[11];

$app\_db[$row] = @dataC[13];

$param1[$row] = @dataC[15];

$param2[$row] = @dataC[17];

$param3[$row] = @dataC[19];

$param4[$row] = @dataC[21];

$row++;

}

$send\_email = 0;

for ($x = 0;$x < $row;$x++)

{

$process\_found = 0;

if ($param1[$x] =~ /pidfile/)

{

open(INFILE,"<$param1[$x]");

$line = <INFILE>;

chop($line);

close(INFILE);

$cmd = "ps -fade | grep -w $line";

}

elsif (length($param4[$x]) > 0)

{

open(INFILE,"<$param4[$x]");

$line = <INFILE>;

chop($line);

close(INFILE);

$cmd = "ps -fade | grep $line";

}

else

{

# Build ps command

$cmd = "ps -fade" . " | grep $process\_nm[$x]";

$server[$x] =~ s/[ ]//g;

if (length($server[$x]) > 0)

{

$cmd = $cmd . " | grep $server[$x]";

}

$task\_type\_cd1[$x] =~ s/[ ]//g;

if (length($task\_type\_cd1[$x]) > 0)

{

$cmd = $cmd . " | grep C$task\_type\_cd1[$x]";

}

$param1[$x] =~ s/[ ]//g;

if (length($param1[$x]) > 0)

{

$cmd = $cmd . " | grep $param1[$x]";

}

} #else

$cmd = $cmd . " | grep -v grep|";

#print "$cmd\n";

open (PROCESS,"$cmd");

while ($inline=<PROCESS>)

{

$process\_found++;

print "PROCESS:$inline\n";

}

close (PROCESS);

$runtime = localtime(time);

if ($process\_found > 1)

{

print OUTFILE "$runtime: Multiple processes running for $process\_nm[$x] $task\_type\_cd1[$x] $param1[$x] for server $server[$x].\n";

$send\_email = 1;

}

elsif ($process\_found < 1)

{

print OUTFILE "$runtime: Process $process\_nm[$x] $task\_type\_cd1[$x] $param1[$x] not running for server $server[$x].\n";

$send\_email = 1;

}

}

$send\_email;

}

sub email

{

#---------------------

# Generate email list.

#---------------------

$query = "SELECT email\_addr

FROM pe\_report..rpt\_assign

WHERE report\_name = 'Job Monitor'";

#print "$query";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

while(@dataC = $sqlC->dbnextrow(1))

{

$email\_list = $email\_list . @dataC[1];

$email\_list = $email\_list . ",";

}

chop($email\_list);

open(MESSAGE,"| mailx -r ccms\@statestreet.com -s \"Message From Job Monitor \" $email\_list < $outfile");

close(MESSAGE);

}

sub Running\_Processes

{

$processing\_time = 0;

while(1)

{

#--------------------------------------------------------

# Change status so that calc/reports don't get processed.

#--------------------------------------------------------

$query = "UPDATE pe\_report..sched\_queue

SET queue\_status\_code = 'W'

WHERE queue\_status\_code = 'R'";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

$query = "SELECT count(\*)

FROM pe\_report..sched\_queue

WHERE queue\_status\_code = 'A'";

$sqlC->dbcmd("$query");

$sqlC->dbsqlexec;

$sqlC->dbresults;

while(@dataC = $sqlC->dbnextrow(1))

{

$processing = @dataC[1];

}

if (($processing == 0) ||

($processing\_time > 8))

{

last;

}

$processing\_time++;

sleep(900);

}

}