

# RUNNING UNIX COMMANDS AND SCRIPTS FROM THE PEOPLESOFT PROCESS SCHEDULER

Prepared By David Kurtz, Go-Faster Consultancy Ltd.

Technical Paper

Version 0.1

Sunday 16 September 2007

(E-mail: [david.kurtz@go-faster.co.uk](mailto:david.kurtz@go-faster.co.uk), telephone +44-7771-760660)

File: process\_scheduler\_shell\_scripts.doc, 16 September 2007

## Contents

Running UNIX Commands and Scripts from the PeopleSoft Process Scheduler .....	1
Introduction .....	2
Process Scheduler Configuration .....	2
Wrapper Script (psft.sh) .....	4
Running the Process .....	7
Appendix .....	8
Demonstration .....	8

## Introduction

This document describes how to run UNIX shell scripts from the Process Scheduler.

At some sites the PeopleSoft system generates interface files that are then delivered to other systems by other shell scripts. These scripts may simply be initiated by the UNIX cron facility. The problem with this is that the scripts run irrespective of whether the PeopleSoft process that generated the interface file ran and completed successfully.

It is possible to use the Process Scheduler to run operating system commands, and to have those command interact appropriately with the generic Process Scheduler functionality.

## Process Scheduler Configuration

### Process Type Definition

First you need to create a new process type, I chose to call it 'Shell Script', that will run a named shell script, *psft.sh*. This wrapper script (see page 4) performs the interaction with the Process Scheduler and it in turn calls the script that is to be executed. The name of the database, the access ID and password, and the process instance are passed to the wrapper. Other parameters will be appended in the individual Process Definition.

The screenshot shows the Oracle Process Scheduler configuration interface. The window title is "Process Types - Go-Faster Consultancy Ltd.". The Oracle logo is visible in the top left. The navigation bar includes "Home", "Worklist", "MultiChannel Console", "Add to Favorites", and "Sign out". The main content area has two tabs: "Type Definition" (selected) and "Type Definition Options".

**Type Definition**

**Process Type:** Shell Script  
**Operating System:** UNIX  
**Database Type:** Oracle

**Details**

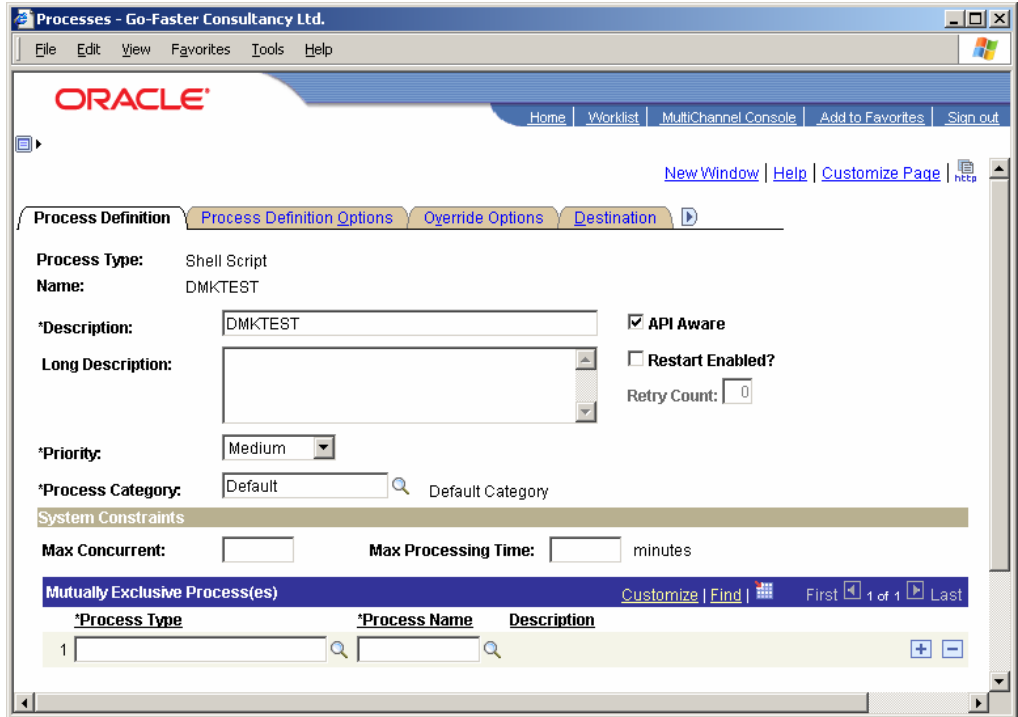
**Description:** Shell Script  
**Generic Process Type:** Other  
**Command Line:** /usr/local/bin/psft.sh  
**Parameter List:** %%DBNAME%% %%ACCESSID%% %%ACCESSPSWD%% %%INSTANCE%%  
**Working Directory:** /usr/local/bin  
**Output Destination:**  
 Restart Enabled

### Process Definition

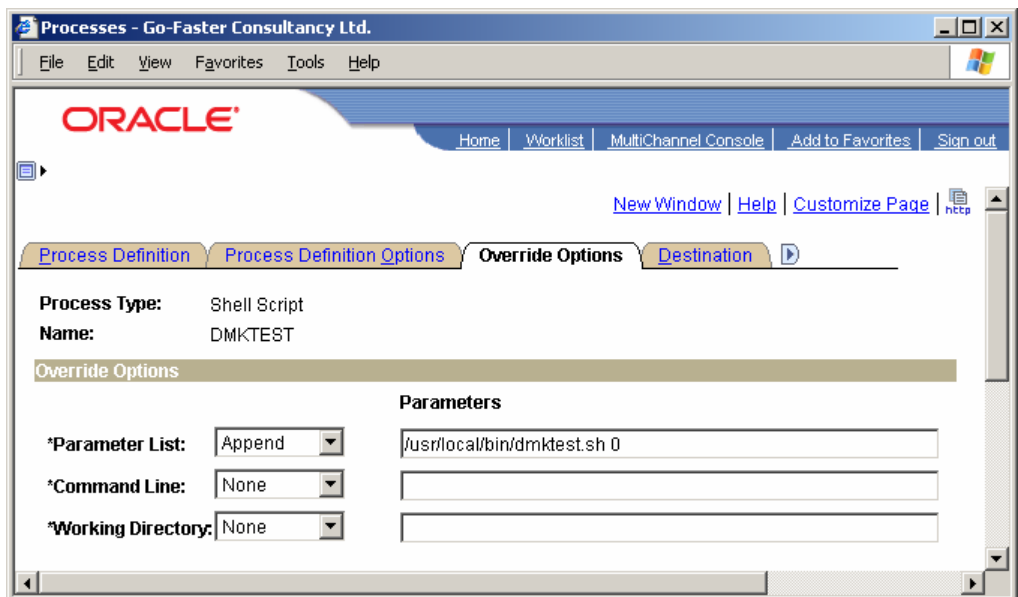
It is necessary to create a Process Definition for each system command or script that is to be executed by the Process Scheduler.

You could simply run the command directly from the Process Scheduler but then it could not be API aware, and the status in the Process Monitor will always be success when it script completes.

When using the wrapper script, the API Aware checkbox should be checked.



The name of the script of system command and any parameters should be APPENDED to the command line defined in the Process Type definition, as shown below.



## Wrapper Script (psft.sh)

This is the wrapper script that is specified in the Process Definition.

```
#!/bin/ksh
#
# Script: psft.sh
#
# Syntax: psft.sh DBNAME ACCESSID ACCESSPSWD PRCSINSTANCE <command line>
# where
# DBNAME is the name of the PeopleSoft database with a corresponding TNS entry
# ACCESSID is the schema containing the PeopleSoft database
# ACCESSPSWD is the password to ACCESSID
# PRCSINSTANCE is the process instance number supplied by PeopleSoft
#
# Purpose: To start Standard UNIX Shell Script from Process Scheduler, and interface with the
PeopleSoft Process Scheduler
#

if [ $# -lt 4 ]; then
    echo "Usage $0: <database name> <access id> <access password> <process instance
number> <command line>"
    exit 1
fi

CONNECT=$2/$3@$11
PRCSINSTANCE=$4
shift 4

#
# Function to set status of API aware process instance
#
function prcsapi2
{
    if [ $# -lt 2 ]; then
        echo "Parameter Error in function $0"
        exit 1
    fi
}

TIMESTAMPCOL=${1}
STATUS=${2}

if [ ${PRCSINSTANCE} -gt 0 ];then
    echo "Setting process request ${PRCSINSTANCE} to status ${STATUS}"
    sqlplus -s /nolog <<!
set termout off echo off feedback off verify off
connect ${CONNECT}
```

<sup>1</sup> The Oracle user ID, password and TNS name for the PeopleSoft database are supplied in the first three parameters. The PeopleSoft Process Instance number is the 4<sup>th</sup> command line parameter for the wrapper. These parameters are then removed with the shift command leaving any other parameters that have been specified.

<sup>2</sup> Function *prcsapi* sets the status on the process request row and updates the appropriate timestamp columns in the Process Scheduler tables.

```

UPDATE psprcsque
SET   runstatus = ${STATUS}
,     sessionidnum = $$3
,     lastupddttm = SYSDATE
WHERE prcsinstance = ${PRCSINSTANCE}
;
UPDATE psprcsrqt
SET   runstatus = ${STATUS}
,     prcsrtncd = ${PRCSRTNCD}
,     continuejob = DECODE(${STATUS},2,1,7,1,9,1,0)4
,     ${TIMESTAMP_COL} = SYSDATE
,     lastupddttm = SYSDATE
WHERE prcsinstance = ${PRCSINSTANCE}
;
COMMIT;
exit
!

    RET=$?
    if [ ! $RET ];then
        echo "SQL*Plus Error Return Code: $?"
    fi
fi
}

#
# Process files in ${PSPRCSLOGDIR}/*
#

function logfiles5
{
#set -x
SEQNUM=0
if [ -d "${PSPRCSLOGDIR}" ]; then
    for FILELIST in ${PSPRCSLOGDIR}/*
    do
        if [ "${FILELIST}" != "${PSPRCSLOGFILE}" ];then
            SEQNUM=$(expr 1 + ${SEQNUM})

            sqlplus -s /nolog <<!
set termout off echo off feedback off verify off
connect ${CONNECT}
INSERT INTO psprcsrqtstfile
(     prcsinstance, seqnum, prcsrqtstfile)
VALUES (${PRCSINSTANCE}
,     ${SEQNUM}

```

---

<sup>3</sup> PSPRCSQUE.SESSIONIDNUM holds the OS process ID of the shell executing the wrapper script.

<sup>4</sup> When the process completes and an end of process status is set (either 9 for success, 3 for failure or 2 for delete) CONTINUEJOB is set to 1, otherwise it is set to 0.

<sup>5</sup> The logfiles function generates entries for all <qwert>

```
,      '${FILELIST}');
COMMIT;
exit
!
          RET=$?
          if [ ! $RET ];then
              echo "SQL*Plus Error Return Code: $?"
          fi
      fi
done
else
    echo "Directory ${PSPRCSLOGDIR} does not exist"
fi
}

#
# Main Execution Starts Here
#

echo $0:$*
date
uname -a
#set
prcsapi beginnttm 76

#Run the command
$*
PRCSRTNCD=$?7

if [ ${PRCSRTNCD} -ne 0 ]; then
    prcsapi enddtm 3 # failure
else
    prcsapi enddtm 9 # success
fi

date
```

---

<sup>6</sup> When the wrapper scripts start it sets the process status to 7 indicate that it is processing.

<sup>7</sup> The return code of the executed script is captured. Later it will be recorded on PSPRCSRQST.PRCSRTNCD.

## Running the Process

The newly defined Process can be run just as any other process is usually run. Any output from the script on the standard output channel is captured by the Process Scheduler and written to a logfile that can then be viewed from the View Log/Trace facility within Process Monitor.

The screenshot shows the Oracle Process Scheduler interface. The title bar reads 'System Process Requests - Go-Faster Consultancy Ltd.'. The main content area is titled 'View Log/Trace Report' and displays the following information:

- Report ID:** 1449
- Process Instance:** 1561
- Name:** DMKTEST
- Process Type:** Shell Script
- Run Status:** Success

Below this, the 'Distribution Details' section shows:

- Distribution Node:** HR89
- Expiration Date:** 09/07/2007

The 'File List' section contains a table with the following data:

Name	File Size (bytes)	Datetime Created
<a href="#">OTH_DMKTEST_1561.log</a>	816	02/07/2007 11:53:30.000000 PDT
<a href="#">dmktest.log</a>	256	02/07/2007 11:53:30.000000 PDT

The 'Distribute To' section shows:

Distribution ID Type	*Distribution ID
User	PS

Two files are shown in the Process Monitor.

- *OTH\_DMKTEST\_1561.log* is the standard output of the script that was captured by the Process Scheduler.
- *dmktest.log* is a file emitted by the called shell script *dmktest.sh* to the reporting directory.

## A P P E N D I X

## Demonstration

### Sample Script

This is the test script called by the Process Definition *DMKTEST*.

```
banner "HelloWorld"8

BASENAME=$(basename $0 .sh)
if [ -d "${PSPRCSLOGDIR}" ]9; then
    echo "This script is running under Process Scheduler"
    cp $0 ${PSPRCSLOGDIR}/${BASENAME}.log10
else
    echo "This script is not running under Process Scheduler"
fi

exit $*
```

### Standard Output

```
/usr/local/bin/psft.sh:/usr/local/bin//dmktest.sh 0
Mon 2 Jul 11:53:15 2007
Setting process request 1561 to status 7
# # # # #
# # ##### # # ##### # # # ##### ##### # #####
# # # # # # # # # # # # # # # # # # # # # #
##### ##### # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # ##### # # # #
# # # # # # # # # # # # # # # # # # # # #
# # ##### ##### ##### ##### ## ## ##### # # ##### #####

This script is running under Process Scheduler
Setting process request 1561 to status 9
Mon 2 Jul 11:53:16 2007
```

---

<sup>8</sup> The banner command is just used here to create some standard output.

<sup>9</sup> Process Scheduler sets an environmental variable PSPRCSLOGDIR. This is the default location for log files. Any log files emitted by the script should be copied to this directory if the variable is set.

<sup>10</sup> The script copies itself to the directory indicated by PSPRCSLOGDIR.