Monitoring with a batch Watchdog Process

Introduction

The Batch Watchdog system detailed is a monitoring system mainly used by administrators. In process control environments where several to many batch jobs are controlling the movement of data for the control of a process (industrial process). The routines are written in VMS C++, Basic, Cobol or Fortran and in real time applications are started periodically by DCL batch routines that resubmit themselves to a batch queue with a future start time which is when the routine is scheduled to run next. This is frequently how VMS routines are used and managed in real time industrial applications.

If a batch job fails and therefore doesn't resubmit itself to a queue, the data files to be processed by the routine are not processed and data is lost. It will resolve itself has a problem with missing data seen by the users.

The Watchdog's purpose is to scan the VMS batch queues every hour and to highlight any missing batch jobs. The scan interval can easily be adjusted by modifying the accompanying Watchdog batch file. It works by comparing the queued batch jobs against a data file which lists the batch job names that should be on the queue and therefore scheduled to run a routine in the future. If there are any missing jobs the Watchdog sends a message to Op Con and to a specified Admin user for each missing job.

The process is important because it gives early warning of any problems that arise highlighting that there is a problem that needs attention. It is particularly useful in real time situations where batch jobs start routines that are moving or processing data into or out of a Database, tidying files, etc.

The Watchdog Command File

The command file BATCH_WATCHDOG.COM is flowcharted in **Diagram 1**, there are several sections to the job.

1) Check Watchdog isn't already executing, set the running lock flag.
2) Calculate the next runtime and submit the watchdog to the batch queue. Code can easily be modified to change the monitoring interval.
3) Create a file containing the running jobs using the output of a 'show queue' command.
4) Convert the running jobs file to lowercase job names only and sort in VMS sort order.
5) Convert the expected jobs file from BATCH_JOBS.DAT so that its in the same form as the real jobs file.
6) Uses two loops, outer loop reads an expected job name, then inner loop, loops through the running jobs list to check it exists.
7) There are a number of possible outcomes.
    a) EOF hit on running jobs list which means this expected batch job isn't running. In which case raise a notification to Op Con and log an entry in the BATCHLOG.LOG file. Go to next step on outer loop (step 5).
    b) Running job exists which is good, move to next step on outer loop (step 5).
8) EOF hit on expected jobs, exit the loops.
9) Tidy files and remove running lock flag.

System Setup

Create a new user WATCHDOG using MCR AUTHORIZE. The command will be similar to:-

ADD WATCHDOG /PASSWORD=<Password>/DEVICE=SYS$USER/DIRECTORY=[WATCHDOG]
/OWNER="WATCHDOG BATCH" /ACCOUNT=SUP

The user where this batch will first be run and WATCHDOG must have the CMKRNL , SYSNAM, SYSPRV and OPER privileges.

Create a Watchdogs directory in SYS$SYSROOT:[WATCHDOGS], this can easily be located anywhere by modifying the code. Copy the BATCH_WATCHDOG.COM and BATCH_JOBS.DAT files into the directory. Change the owner of the directory and files to be the WATCHDOG user, the command will be similar to :-

Set security/owner=WATCHDOG <FILENAME>

Create a WATCHDOGS batch queue.

INIT/QUEUE/BATCH/START WATCHDOGS

Change the default directory to SYS$SYSROOT:[WATCHDOGS] and start the watchdog command job.

SET DEF SYS$SYSROOT:[WATCHDOGS]
@BATCH_WATCHDOG

Monitoring the Batch Queues

The way the real time processes are started will be different for each situation. The watchdog should not be started until all the batch files have executed and resubmitted themselves to a batch queue, see **Figure 1**, otherwise notification messages will be sent for any missing jobs on the queues.

```
Reply received on VMS923 from user SYSTEM at _TNA261:   09:58:35
 EOF HIT BATCH JOB test4 MISSING FROM BATCH QUEUE

User SYSTEM has been notified on VMS923 (2 terminals).
$ SHO QUEUE/ALL/BATCH
Batch queue TEST, idle, on VMS923::

  Entry  Jobname         Username             Status
  -----  -------         --------             ------
     19  test2           WATCHDOG             Holding until 18-NOV-2025 10:10:00.00
     21  test3           WATCHDOG             Holding until 18-NOV-2025 10:10:00.00

Batch queue TEST2, idle, on VMS923::

  Entry  Jobname         Username             Status
  -----  -------         --------             ------
     20  test1           WATCHDOG             Holding until 18-NOV-2025 10:10:00.00

Batch queue WATCHDOGS, idle, on VMS923::

  Entry  Jobname         Username             Status
  -----  -------         --------             ------
     22  BATCH_WATCHDOG  WATCHDOG             Holding until 18-NOV-2025 10:50:00.00
$
```

Figure 1

Once the BATCH_WATCHDOG is running and resubmitting itself to the WATCHDOGS queue it will monitor the batch queues and report any missing jobs hourly, see **Figure 2**.

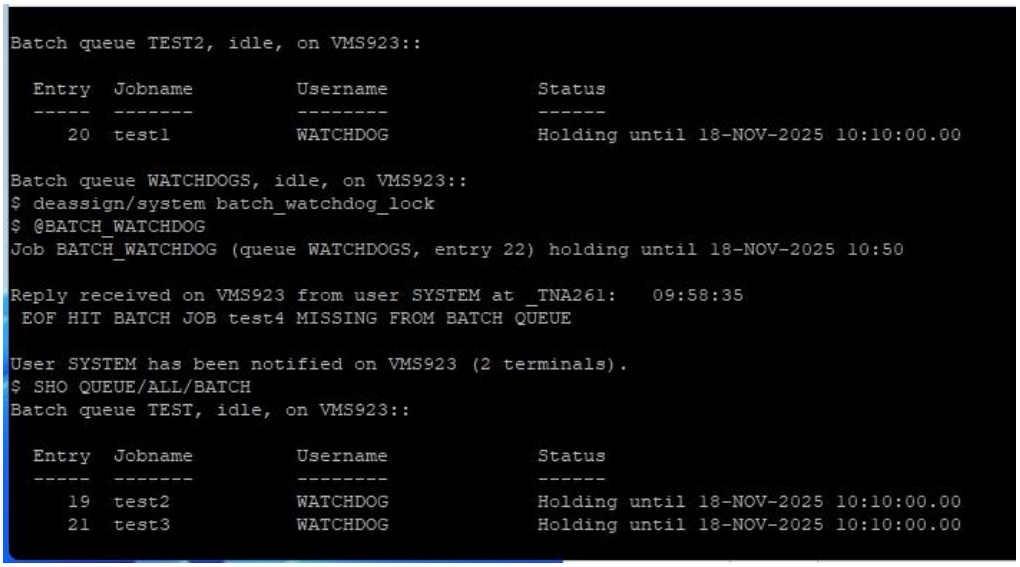
Batch queue TEST2, idle, on VMS923::
Entry Jobname Username Status
----- ------- -------- ------
20 test1 WATCHDOG Holding until 18-NOV-2025 10:10:00.00
Batch queue WATCHDOGS, idle, on VMS923::
$ deassign/system batch_watchdog_lock
$ @BATCH_WATCHDOG
Job BATCH_WATCHDOG (queue WATCHDOGS, entry 22) holding until 18-NOV-2025 10:50
Reply received on VMS923 from user SYSTEM at _TNA261: 09:58:35
EOF HIT BATCH JOB test4 MISSING FROM BATCH QUEUE
User SYSTEM has been notified on VMS923 (2 terminals).
$ SHO QUEUE/ALL/BATCH
Batch queue TEST, idle, on VMS923::
Entry Jobname Username Status
----- ------- -------- ------
19 test2 WATCHDOG Holding until 18-NOV-2025 10:10:00.00
21 test3 WATCHDOG Holding until 18-NOV-2025 10:10:00.00

Figure 2

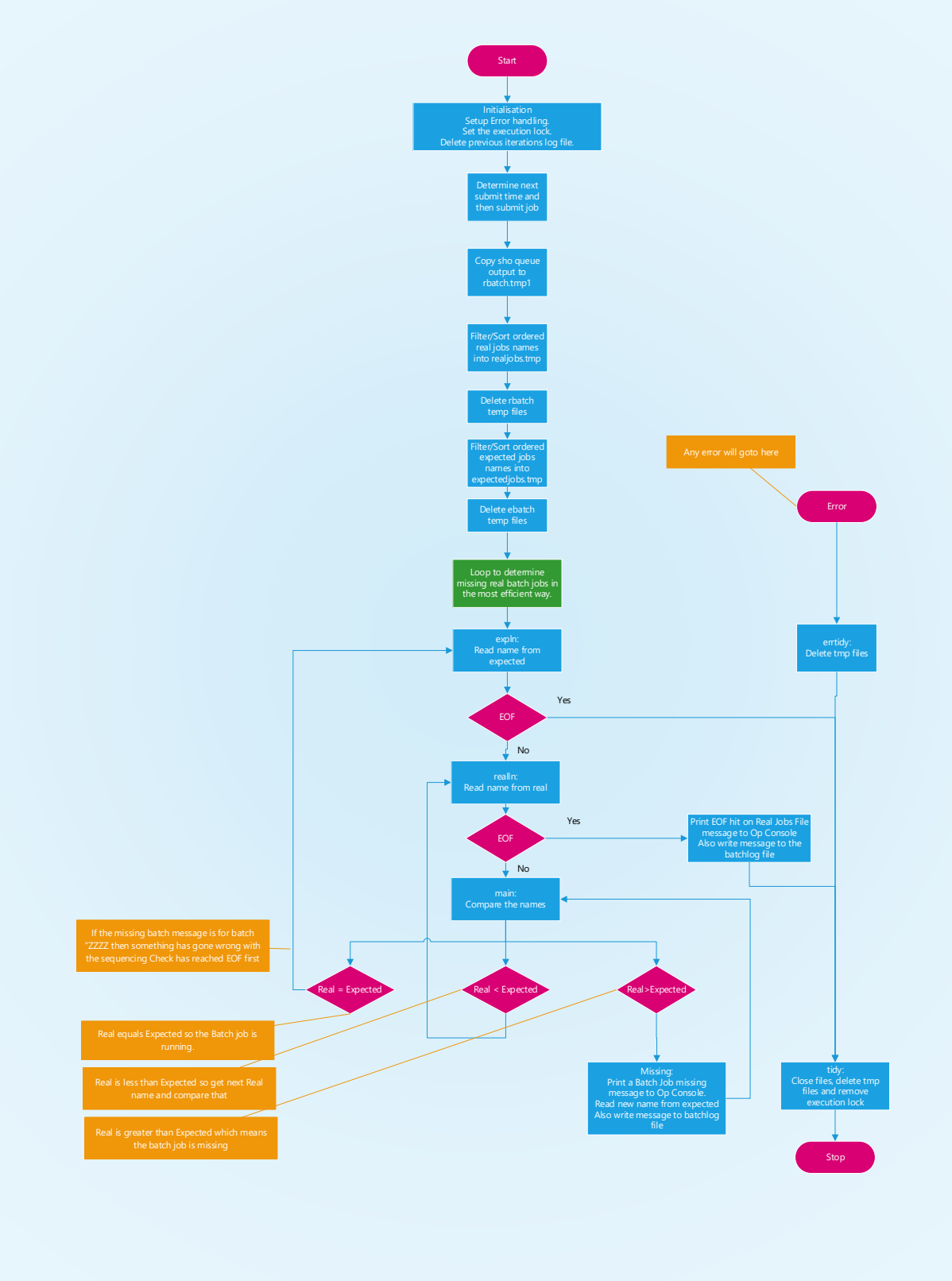

Diagram 1 – Proposed Batch_Watchdog Process

Appendix A

```
$! BATCH_WATCHDOG
$!
$!   Checks all Queues for the presence of all jobs in BATCH_JOBS.DAT
$!
$! AMENDMENT RECORD:
$!
$!  Created by                          CW Walker 28-May-2020
$!
$!
$!
$!  --Initialisation Routine --
$!
$! ---If already running exit ---
$!
$ set def sys$sysroot:[watchdogs]
$ if f$trnlnm("batch_watchdog_lock") .NES. ""
$ then
$   REQUEST/TO=CENTRAL "BATCH_WATCHDOG ALREADY RUNNING"
$   REPLY/USERNAME=SYSTEM "BATCH_WATCHDOG ALREADY RUNNING"
$   exit
$ endif
$!
$ define/system batch_watchdog_lock running
$!
$ on warning then goto errtidy
$!
$! --- Delete previous logfile if it exists ---
$!
$ if f$search("batchlog.log") .NES. "" then delete/NOCONF batchlog.log;*
$!   ----------------------------------------
$! -- Submit a new iteration to the WATCHDOG Queue, To run in 1 hour --
$! --- Example iterates every hour, but could be modified for any time interval ---
$!
$ thour    = f$cvtime(,,"hour")
$!
$ if thour .eq. 23
$ then
$    submit_time = "TOMORROW+00:50"

$ else
$    thour = thour + 1
$    thour = "'"thour'" + ":50"
$    submit_time = "TODAY+" + "'"thour'"
$ endif
$!
$!!
$ SUBMIT/USER=WATCHDOG /QUEUE=WATCHDOGS/AFTER="'"submit_time'"/NOPRINT
SYS$SYSROOT:[WATCHDOGS]BATCH_WATCHDOG
$ open/write batchlog batchlog.log
$ write batchlog "Batch_Watchdog submitted to queue"
$!  ----------------------------------------------
$!  -- Create a file containing the real running jobs --
$!
$ sh queue/batch/all/output=rbatch.tmp1
$ write batchlog "Running jobs file created"
$!  ----------------------------------------------
```

```
$!  -- Filter the Real jobs file to contain only job --
$!  -- names in lowercase and VMS sort the names     --
$!
$ open/read  infile  rbatch.tmp1
$ open/write outfile rbatch.tmp2
$read1:
$ read/end=end1 infile rinrec
$ if f$extract(6,1,rinrec) .lts. "0" then goto read1
$ if f$extract(6,1,rinrec) .gts. "9" then goto read1
$ routrec = f$extract(9,16,rinrec)
$ routrec = f$edit(routrec,"lowercase")
$ routrec = f$edit(routrec,"TRIM")
$ write outfile routrec
$ goto read1
$end1:
$ close infile
$ close outfile
$!
$ sort rbatch.tmp2 realjobs.tmp
$!  ---------------------------------------------
$!  -- Tidy files so there is only a sorted lowercase  --
$!  -- real jobs list in RealJobs.tmp                --
$!
$ if f$search("rbatch.tmp*") .NES. "" then delete/NOCONF rbatch.tmp*;*
$!
$!  ---------------------------------------------
$!  -- Filter and sort the Expected job names file so  --
$!  -- that it has the same form has the realjobs file --
$!
$ open/read  infile SYS$SYSROOT:[watchdogs]batch_jobs.dat
$ open/write outfile ebatch.tmp1
$read2:
$ read/end=end2 infile einrec
$ if f$extract(0,1,einrec) .eqs. "!" then goto read2
$ eoutrec = f$extract(0,16,einrec)
$ eoutrec = f$edit(eoutrec,"lowercase")
$ eoutrec = f$edit(eoutrec, "TRIM")
$ write outfile eoutrec
$ goto read2
$end2:
$ close infile
$ close outfile
$!
$ sort ebatch.tmp1 expectedjobs.tmp
$!  ---------------------------------------------
$!  -- Tidy files so there is only a sorted lowercase  --
$!  -- expected jobs list in expectedjobs.tmp --
$!
$ if f$search("ebatch.tmp1") .NES. "" then delete/NOCONF ebatch.tmp1;*
$!
$!  ---------------------------------------------
$! --Loop to determine missing jobs --
$! --uses realjobs and expectedjobs file --
$! --Double loop outer real inner expected --
$! -- Open the files --
$!
$ open/read real realjobs.tmp
$ open/read expected expectedjobs.tmp
```

```
$ open/write batchlog batchlog.log
$!
$! --Double loop outer real inner expected --
$ expln:
$ read/end=expEOF expected expbuff
$ realln:
$ read/end=realEOF real realbuff
$!
$ main:
$! --Loop depending upon compare of sorted files --
$!
$! --real batch job is found, so goto next expected job --
$ if realbuff .EQS. expbuff then goto expln
$!
$! --real batch job is further down the list, so get --
$! --next real job --
$ if realbuff .LTS. expbuff then goto realln
$!
$! --real batch job is missing, so send a message --
$ if realbuff .GTS. expbuff then goto missing
$!
$!  ----------------------------------------
$! --Missing real batch job, so move to next --
$! --Expected name to compare against existing --
$! --real name, oh and send missing message --
$ missing:
$ MESSAGE = "BATCH JOB ''expbuff' MISSING FROM BATCH QUEUE"
$ write batchlog MESSAGE
$ REQUEST/TO=CENTRAL " ''MESSAGE'"
$ REPLY/USERNAME=SYSTEM " ''MESSAGE'"
$ read/end=expEOF expected expbuff
$ goto main
$!  ----------------------------------------
$! --EOF reached on expected file --
$ ExpEOF:
$!
$ MESSAGE = "EOF HIT ON EXPECTED JOBS, SCAN COMPLETED"
$ write batchlog MESSAGE
$ REQUEST/TO=CENTRAL "'MESSAGE'"
$ REPLY/USERNAME=SYSTEM " ''MESSAGE'"
$ goto tidy
$!
$!  ---------------------------------------
$! --EOF reached on real file --
$realEOF:
$! MESSAGE = "EOF HIT ON REAL JOBS FILE"
$ MESSAGE = "EOF HIT BATCH JOB ''expbuff' MISSING FROM BATCH QUEUE"
$ write batchlog MESSAGE
$ REQUEST/TO=CENTRAL "'MESSAGE'"
$ REPLY/USERNAME=SYSTEM " ''MESSAGE'"
$!
$ goto tidy
$!
$!  -------------------------------------
$!  --Tidy everything because an error --
$!  -- occurred                  --
$ errtidy:
$ if f$search("rbatch.tmp*") .NES. "" then delete/NOCONF rbatch.tmp*;*
```

```
$ if f$search("ebatch.tmp*) .NES. "" then delete/NOCONF ebatch.tmp*;*
$!  ---------------------------------
$ tidy:
$ close expected
$ close real
$ close batchlog
$! --- keep the sorted files, testing
$! exit
$!
$ if f$search("realjobs.tmp") .NES. "" then delete/NOCONF realjobs.tmp*;*
$!
$ if f$search("expectedjobs.tmp") .NES. "" then delete/NOCONF expectedjobs.tmp*;*
$!
$ deassign/system batch_watchdog_lock
$!
$ exit
```

Appendix B

```
! BATCH_JOBS.DAT
!
! NOTE: Job names can be in any order the watchdog batch sorts them
! ======================================================
!
TEST1
TEST2
TEST3
TEST4
!TEST5        -NO
TEST6
```