

# Commit Trigger Example

A configurable number of revisions

By Richard E. Baum, Perforce Software



# Introduction

- For binary files, Perforce offers the ability to store all revisions or only the last one.
- These choices are not always the most desirable.
  - Many large revisions – disk space problems
  - Only last revision – revert is impossible
- A configurable number of revisions would be helpful!



# Introduction

- One solution, a post-submit trigger
  - Executes after changelist submission
  
- Bourne shell
  - Easy to implement
  - Easy to understand
  - Works on most all platforms



# What it does

- After a changelist is submitted
  - Files in changelist are checked sequentially
    - Match path for the trigger
    - Proper storage type (binary/revision)
  - Revisions of each file are sorted in time order
  - Large revision (over 10 blocks) are counted
  - Revisions over target are replaced with stub
  - MD5 hash is fixed with `p4 verify -v`



# Limitations

- 2004.2 or later Perforce server
- Special characters in path can cause quoting problems
  - Some characters known to cause problems are:  
% ( ) # \$ \*
- Computations are based on filesystem dates
  - Outside modification of file tree will cause undesired results
- Purging the source of a lazy copy will result in “BAD!” messages from `p4 verify`



# Input

- The trigger requires two arguments
  - Changelist number
  - Number of revisions/file to keep



# Script internal setup

<b>Variable</b>	<b>Value</b>
MSG	Message for stub file
DEPOT_PATH	Path the trigger matches
P4ROOT	Absolute path of the versioned files
LOG	Where to log output
BLOCKS	Disk blocks required to count as a large file (Often, 1kb = 1 block)
P4	Path to the Perforce client executable
PORT	Perforce port for server connections
CLIENT	Perforce client specification to use



# Perforce setup

## ➤ Trigger specification line:

```
rev_purge commit //depot/bin/...  
    "/path/purge_trigger.sh %changelist% 10"
```



# The script

```
MSG="This file was removed at `date` to save space!"
DEPOT_PATH="//depot/bin/..."
ROOT=/perforce/purge
LOG=/perforce/purge/purge.log
BLOCKS=10
P4=/usr/team/reb/test/p4
PORT=7890
CLIENT=reb_play
CHANGE=$1
REVS2KEEP=$2
date >> $LOG
P4="$P4 -p $PORT -c $CLIENT"
```



```

# Strip off hash, ver & other non-filename parts.
# Dereference spaces to "%" - keep paths a single "for" arg
for FILE in ` $P4 files $DEPOT_PATH@=$CHANGE | \
    sed -e 's/\(.*\)\#[0-9]* - .*$/\1/' -e 's/ /%/g' `
do
    # Undo the dereference and remove leading //
    FILE="`echo $FILE | sed -e 's/%/ /g' -e 's|^//|'|`"
    #
    # If a ,d file it's a file/revision file else ignore.
    if [ -d $ROOT/${FILE},d ]
    then
        #Do each file in the revision dir in time order.
        THISREV=0
        for REV in `ls -1t $ROOT/${FILE},d/1.*.gz`
        do
            # Figure out if it's big or small, count, process...
            SIZE=`ls -s $REV | sed -e 's/\(.*[0-9]*\)\ .*/\1/'`
            if [ $SIZE -gt $BLOCKS ]
            then

```



```

let THISREV=$THISREV+1
if [ $THISREV -gt $REVS2KEEP ]
then
    # Do the dirty work - delete the big file!
    echo "$MSG" | gzip -9 > $REV
    #
    # Calculate version
    VER=`echo $REV | sed -e 's|/. *1\. \([0-9]*\) \.gz|\1|'`
    #
    # Fix up the MD5 hash
    p4 verify -v "$FILE#$VER,$VER"
    echo "$REV BIG    $THISREV DELETED" >> $LOG
else
    echo "$REV BIG    $THISREV of $REVS2KEEP" >> $LOG
fi
fi
done
fi
done
exit 0
# END

```



# Sample run - before

- Prior to running for the first time, 11 revisions exist:

```
-rw-r--r-- 1 reb team 4288726 Feb 26 12:43 1.1.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:45 1.2.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:46 1.3.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:47 1.4.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:48 1.5.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:52 1.6.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:53 1.7.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:53 1.8.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:53 1.9.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:54 1.10.gz
-rw-r--r-- 1 reb team 4288726 Feb 26 12:56 1.11.gz
```



# Sample run - After

- After the trigger is in place, revision 12 is submitted.  
The program log shows:

Sat Feb 26 12:57:51 PST 2005

```
/perforce/purge/depot/bin/binfile,d/1.12.gz BIG 1 of 10
/perforce/purge/depot/bin/binfile,d/1.11.gz BIG 2 of 10
/perforce/purge/depot/bin/binfile,d/1.10.gz BIG 3 of 10
/perforce/purge/depot/bin/binfile,d/1.9.gz BIG 4 of 10
/perforce/purge/depot/bin/binfile,d/1.8.gz BIG 5 of 10
/perforce/purge/depot/bin/binfile,d/1.7.gz BIG 6 of 10
/perforce/purge/depot/bin/binfile,d/1.6.gz BIG 7 of 10
/perforce/purge/depot/bin/binfile,d/1.5.gz BIG 8 of 10
/perforce/purge/depot/bin/binfile,d/1.4.gz BIG 9 of 10
/perforce/purge/depot/bin/binfile,d/1.3.gz BIG 10 of 10
/perforce/purge/depot/bin/binfile,d/1.2.gz BIG 11 DELETED
/perforce/purge/depot/bin/binfile,d/1.1.gz BIG 12 DELETED
```



# Sample run - After

- The versioned file tree now looks like this, with the first two revisions truncated:

```
-rw-r--r--  1 reb  team          108 Feb 26 13:08 1.1.gz
-rw-r--r--  1 reb  team          108 Feb 26 13:08 1.2.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:46 1.3.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:47 1.4.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:48 1.5.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:52 1.6.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:53 1.7.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:53 1.8.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:53 1.9.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:54 1.10.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:56 1.11.gz
-rw-r--r--  1 reb  team    4288726 Feb 26 12:57 1.12.gz
```



# Sample run - After

- A listing of one of the stub file revisions shows the new file content:

```
p4 print //depot/bin/p4d#2
//depot/bin/p4d#2 - edit change 2 (xbinary)
This file was automatically removed at Sat Feb 26
13:08:27 PST 2005 in order to save disk space!
```



# Sample run - After

## ➤ Verify reports no errors:

```
:  
//depot/bin/p4d#12 - edit change 12 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#11 - edit change 11 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#10 - edit change 10 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#9 - edit change 9 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#8 - edit change 8 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#7 - edit change 7 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#6 - edit change 6 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#5 - edit change 5 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#4 - edit change 4 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#3 - edit change 3 (xbinary)3C8753BF7665650980  
//depot/bin/p4d#2 - edit change 2 (xbinary)83A33300DD2AA3C1FE  
//depot/bin/p4d#1 - add change 1 (xbinary)83A33300DD2AA3C1FE
```



# Future improvements

- Archive files instead of deleting them
  - Offline storage
  - Nearline storage
  
- Notify Perforce server of archive location
  - Journal patch of db.rev



# Conclusion

- Use this script is a guide
- Do not use it out-of-the-box
- It has limitations
- It deletes data
- Use with **EXTREME CARE!**

