Extending the Dollar Universe System

by

Gary F. Alderson

Aldersoft.CA

Extending the Dollar Universe System

- Agenda
  - Finding Hidden Gems in Dollar Universe
  - The $U Command Line Utilities
  - U_BATCH, U_ANTE_PROC, U_POST_PROC
  - Did you know about $U Runbooks ?
  - Using BAT, SH and PERL to build Gems
  - Some Simple Examples
  - Questions and Hopefully Answers

Finding Hidden Gems in Dollar Universe

- $U is filled with many rich facilities and functions that are not generally discussed.
- Using these Gems, one can create your own add-ons and extensions to customize and extend the product.
- The intent here is to provide an overview to show you where these Gems are and a brief idea on how to get started using them.

The $U Command Line Utilities

- Source: “Dollar Universe Command Interface”
- Almost everything that can be done through the Dollar Universe Console can be done using these command line functions.
- By adding these commands into UPROCS, you can perform functions dynamically and even develop your own commands.

Parameter Setting Commands

- Source: Dollar Universe Command Manual
- Parameter Setting Commands are designed to add, duplicate, update, delete, show, list, distribute, transfer, extract, insert, unlock and source version control the various $U Objects and their attributes.
- In essence, you have the ability to manage the entire repository of $U objects programmaticality
- Ideas the come to mind:
  - Synchronize distributed repositories with a master repository.
  - Promotion of objects from development to production.
  - Produce various repository management reports.

Operations Commands

- Source: Dollar Universe Command Manual
- Operations commands control the $U operations environment. Eg. Future Launches, Executions, Statistics, Events in terms of adding, updating, purging, showing, forecasting, etc.
- In essence, you have the ability to manage the operations environment.
- Ideas that come to mind:
  - End of day reporting
  - Historical estimates
  - Audit Reporting and Queries

Batch Management Commands

- Source: Dollar Universe Command Manual
- Batch Managements commands are used to submit and control task executions by triggering provoked tasks, enforce execution time limits by killing tasks, take exception steps if time limits are exceeded.
- In the latter case, you will probably want to use uxalrjob, which is more flexible in terms of parameters.
- In the case of provoked tasks, this can be a blessing for dynamically managing exception situations such as. “Run report x only if there was a cheque issued for over $10,000.”

CL Commands

- Source: Dollar Universe Command Manual
- CL Commands must be run from the executing UPROC and do many handy things for you.
- For example: passing variables to the next UPROC, preventing the running UPROC from being canceled during critical steps, identifying steps for checkpoint/restart, issuing severity and information viewable on consoles, generating log messages, etc.
- Ideas that come to mind:
  - Let me count the ways ...

U_BATCH, U_ANTE_UPROC and U_POST_UPROC

- Source: $UXMGR and $UXEXE directories (read the scripts' code)
- See Also: uxsetenv
- $U setup by Professional Services has U_ANTE_PROC and U_POST_PROC added called by U_BATCH before and after each UPROC is run.
- Let's examine these further.

$UXEXE/u_batch

- $UXEXE/u_batch is called to run every UPROC during the launch process. (In Windows this is %UXEXE %\u_batch.bat)
- First u_batch sets up the UPROC's environment.
- Then $UXMGR/U_ANTE_PROC is run if it exists via sourcing (run under the same shell environment).
- The UPROC is then run using eval (Windows via call) to trap errors and messages.
- Next $UXMGR/U_POST_PROC is run if it exists via sourcing. (run under the same shell environment).
- Finally it runs the termination end of job routine.

Handy Environment Variables

Variable	Set By	Description	Variable	Set By	Description
S_CODEXP	Launcher	App Type | Area	U_FMT_DATE	uxsetenv	Date Format
S_CODPROF	uxsetenv	User Profile	U_TMP_PATH	uxsetenv	Node Temp Dir
S_CODSESS	Launcher	Session Name	UX_FAILURE_STRING	Launcher	Search Log for failure
S_CODUG	Launcher	Management Unit	UX_SUCCESS_STRING	Launcher	Search Log for success
S_ESPEXE	u_batch	Execution Area	UXA{AP|IN|SI|EX}	uxsetenv	Area Action Dir
S_NOEUD	uxsetenv	Node Name	UXD{AP|IN|SI|EX}	uxsetenv	Area Data Dir
S_NUMPROC	Launcher	Execution Uproc No	UXDIR_ROOT	uxsetenv	Company Root Dir
S_NUMSESS	Launcher	Execution Session No	UXEXE	uxsetenv	$U Executable Dir
S_PROCEXE	Launcher	Basic UPROC Name	UXL{AP|IN|SIEX}	uxsetenv	Area Logs Dir
S_SIGCELL	Launcher	DQM Queue	UXMGR	uxsetenv	$U Maintenance Dir
S_SOCIETE	uxsetenv	Company	UXP{AP|IN|SI|EX}	uxsetenv	Area Scripts Dir
S_TIMEOUT	uxsetenv	Disconnect Time	UXVERBOSE	externally	Trace scripts {true|false}
S_U_LANGUE	uxsetenv	Current Language	RESEXE	u_batch	UPROC Status Code
S_UPROC	Launcher	FQ UPROC Name	FILE_LOG	upostproc	UPROC Log File
S_USERNAME	Launcher	Execution Userid	… and many more		Local to U_POST_PROC

$UXMGR/U_ANTE_PROC

- Once the UPROC environment has been initialized, u_batch sources (Windows calls) U_ANTE_PROC
- U_ANTE_PROC as installed does the following:
  - Differentiates processing based on Areas, EG. In Area=APP it causes the UPROC to "sleep 10"instead of executing.
  - Sets run alarms (uxalrjob and uxspvjob)
  - Sets run terms (uxsurjob)
  - Loads "Environment" files based on naming conventions:
    - main.env, {Area}.env, {Area}_node_{Node}.env
    - {Area}_prj_{ProjectID}.env, {Area}_app_{Team}.env
    - {Area}_{AppType}.env and {Area}_mu_{MU}.env

$UXMGR/U_POST_PROC

- Once the UPROC completes, u_batch sources U_POST_PROC
- U_POST_PROC as installed does the following:
  - Parses the UPROC's log file for any defined Success or Failure strings using standard “grep”, hence some regular expressions ;-)
  - Manages any abort status …
  - Optionally, changes the permissions of the log file via
    - `[[ -f $FILE_LOG ]] && chmod 660 $FILE_LOG >/dev/null 2>&1`
    - This is a good idea if your people use umasks other than 007 as it will set permissions so the log can be viewed by the consoles.
  - Optionally, emails or pages upon failure, beyond the UPROC definitions.
  - You can also code in whatever else you might like to do ;-)

Do you know about $U Runbooks ?

- Within U_POST_PROC is code which will search for an appropriate “Runbook” for the UPROC which has aborted, and if found its contents will be displayed in the message log.
- The intent is to avoid searching through documentation and provide restart / recovery instructions to whomever is responsible by simply reading these in the message log.

$U Runbooks

Inside U_POST_PROC exists the following code:

```
#####################################################
# Manage abort status
#####################################################
If  [[ ${RESEXE_LOCAL:=0} -ne 0 ]];then
    Message "Return Code = ${RESEXE_LOCAL}"
    file="${UXDIR_ROOT}/runbooks/${S_PROCEXE}_runbook.txt"
    If  [[ -f ${file} ]];then
        while read line;do
            ${UXEXE}/uxset msg "${line}"
        done < ${file}
    else
        ${UXEXE}/uxset msg \
            "WARNING: No runbook defined for job ${S_PROCEXE}"
    fi
fi
```

Hence if a file exists in the runbooks directory under the installation directory, and it is named YourUPROC_runboook.txt, then it gets displayed in the message log.

A $U Runbook Example

Contents of $UXDIR_ROOT/runbooks/EE_AKABOOM_runbooks.txt

```
RUNBOOK for EE_AKABOOM
======================

PURPOSE:
========
This UPROC is design to explode and simulate a UPROC failure.

RECOVERY PROCEDURE:
===================
There is absolutely nothing that you can or should do about this.
I am just putting in a entry to show how to write a runbook.

ACTIONS TO TAKE IF YOU CANNOT RECOVER:
======================================
All ye who enter here, abandon all hope.

END OF RUNBOOK FOR EE_AKABOOM
=============================
```

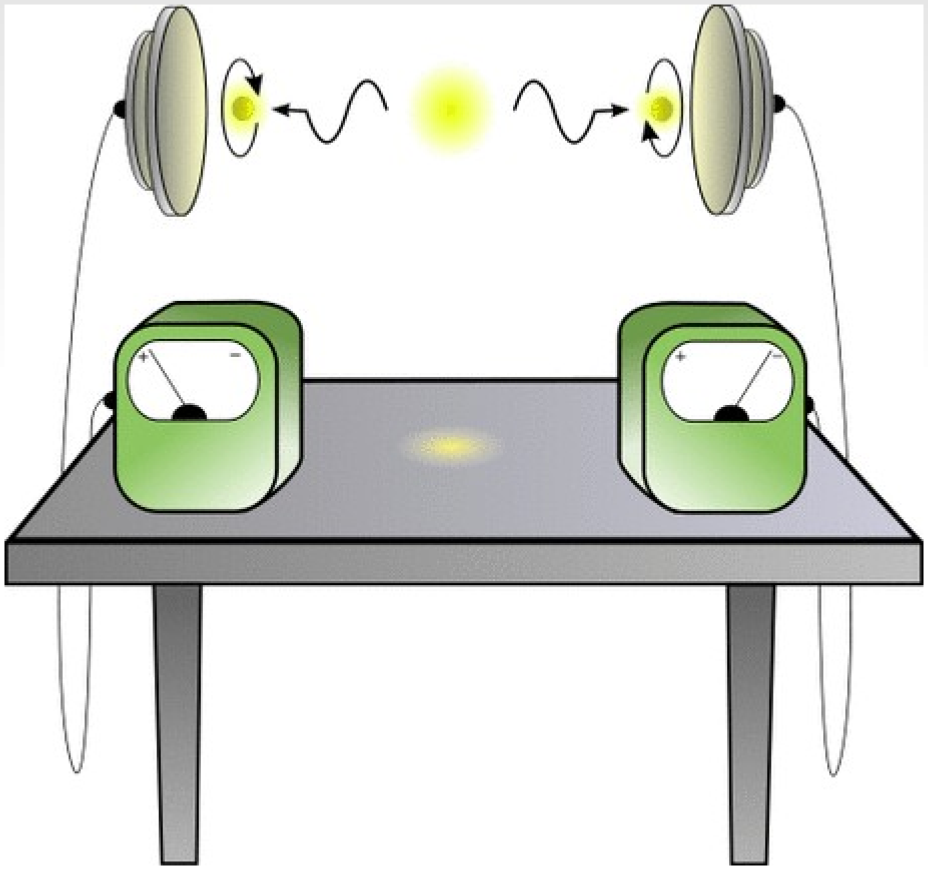
$U Runbook History Log

```
----------------------------------------------------------
End of uproc processing
----------------------------------------------------------
Return Code = 16
RUNBOOK for EE_AKABOOM
======================
PURPOSE:
========
This UPROC is design to explode and simulate a UPROC failure.
RECOVERY PROCEDURE:
===================
There is absolutely nothing that you can or should do about this.
I am just putting in a entry to show how to write a runbook.
ACTIONS TO TAKE IF YOU CANNOT RECOVER:
======================================
All ye who enter here, abandon all hope.
END OF RUNBOOK FOR EE_AKABOOM
=============================
Exit code :  16
2010/09/26 01:34:40 *** TASK ENDED ABNORMALLY  ***
*** TOTAL *** - Duration : 00.00:00:05.00  - CPU : 00.00:00:00.92
              - PGF   : ????  - DIO : ????  - BIO : ????
2010/09/26  01:34:40 NOTIFICATION_EMAIL REQUEST OK - Aborted
Gary_Alderson@UManitoba.CA;
 *** Uproc ABORTED. Completion Instructions were not executed ***
```

Using BAT, SH and PERL to build Gems

- $U uses .bat language for the Windows platform and Korn Shell (ksh) for the UNIX and (bash) for Linux platforms.
- ORSYP has done wonders with these languages.
- I prefer to develop solutions using a common language. My preference is PERL, available on most platforms.
- ORSYP also provides the building blocks for C.

Some Simple Examples



- A wrapper for uxset msg.
- An alternative to uxset vars for passing variables.
- End of UPROC cleanup.

1. A Wrapper for uxset msg

Why ? To be proactive to change and to catch common errors.

```
#-----------------------------------------------------------------
# Function...........: dc_uproc_message
# Author ............: Gary F. Alderson
# Date-Written.......: 2008-06-23
# Purpose............: Send a message to the Dollar Universe
#                      Execution History and JobLog.
#-----------------------------------------------------------------
dc_uproc_message() {
    _DC_MSG="$1"
    echo $_DC_MSG
    $UXEXE/uxset msg "$_DC_MSG"
    unset _DC_MSG
}
```

1. A Wrapper for uxset msg

This time for Windows … dc_uproc_message.bat

```
@echo off
rem ---------------------------------------------------------
rem Function...........: dc_uproc_message
rem Author ............: Gary F. Alderson
rem Date-Written.......: 2008-06-23
rem Purpose............: Send a message to the Dollar Universe
rem                      Execution History and JobLog.
rem ---------------------------------------------------------
if  not defined UXEXE        goto noenvset
call %UXEXE%\uxset msg "%1 "
echo %1
goto end
:noenvset
echo The $Universe environment has not been set.
echo Please make sure you add: call UXSETENV.bat to your UPROC.
set DC_MAXCC=%DC_FAILURE_LIMIT%
:end
```

2. An alternative to uxset vars for passing variables

- Why ?
  - uxset vars must be done in every UPROC in a session, thus you need to be aware of both structure and you must define every variable.
  - What happens when UPROCs are shared between multiple sessions which pass different variables ?
- Solution:
  - Pass variables to the session header.
  - Create an xml file of variables that belong to the session.
  - Retrieve as required, and use environment variables.

Setting Session Variables in the Header

```
#!/bin/sh
#----------------------------------------
# Script...........: dp_hepftp
# Author...........: Matt Dunlop
# Date-Written.....: 2009-09-23
# ...
#----------------------------------------
. /local/adminsys/bin/dc_uproc_functions
dc_uproc_start

DP_SOURCE_FILE=/home/adpc/cservap/at/du_test/my_f*.txt
EPRINT_USERID=jason_bourne
EPRINT_PSWD=very_secret
DP_FTP_HOST=$DP_FTP_HOST # Passed as a variable perhaps

dc_uproc_set_session_variables "DP_SOURCE_FILE EPRINT_USERID
   EPRINT_PSWD DP_FTP_HOST"

dc_uproc_end
```

Retrieving Session Variables in the Application UPROC

```
#!/bin/sh
#------------------------------------------
# Script............: DP_AEPFTPM
# Author............: Matt Dunlop
# Date-Written.....: 2009-09-23
# ...
#------------------------------------------
. /local/adminsys/bin/dc_uproc_functions
dc_uproc_start

dc_uproc_get_session_variables
echo “Parameters passed:”
echo “DP_SOURCE_FIlE : $DP_SOURCE_FILE”
echo “EPRINT_USERID  : $EPRINT_USERID”
echo “EPRINT_PSWD    : $EPRINT_PSWD”
echo “DP_FTP_HOST    : $DP_FTP_HOST”
...
dc_uproc_end
```

Setting Session Variables

```
dc_uproc_set_session_variables() {
if  [ -z "$S_CODSESS" ]; then
    dc_uproc_message "dc_uproc_set_session_variables can only be used in sessions"
    dc_uproc_message "no xml file will be produced, thus later errors could occur"
    return 4
fi
_SESSION_XML_FILE="${U_TMP_PATH}/${S_ESPEXE}_${S_CODSESS}_${S_NUMSESS}.xml"
_SESSION_VARS="$*"
if  [ -z "$_SESSION_VARS" ]; then
    dc_uproc_message "dc_uproc_set_sessions_variables was not passed any parmeters"
    dc_uproc_message "no xml file will be produced, thus later errors could occur"
    return 5
fi
_TD=`date +%Y-%m-%d\ %H:%M:%S`
echo "<?xml version=\"1.0\"?>" > $_SESSION_XML_FILE
echo "<!-- Company:$S_SIGSOC Area:$S_ESPEXE Session:$S_CODSESS -->" >>
   $_SESSION_XML_FILE
echo "<!-- SessionVersion:$S_NUMVERSESS SessionNo:$S_NUMSESS -->" >> $_SESSION_XML_FILE
echo "<!-- CreatedByUPROC: $S_UPROC At:$_TD -->" >> $_SESSION_XML_FILE
echo "<defines>" >> $_SESSION_XML_FILE
echo "    <session>" >> $_SESSION_XML_FILE
for _V in $_SESSION_VARS; do
    eval _VV=`echo "\$"$_V`   # Dereference a variable name to its value
    echo "        <$_V>$_VV</$_V>" >> $_SESSION_XML_FILE
done
echo "    </session>" >> $_SESSION_XML_FILE
echo "</defines>" >> $_SESSION_XML_FILE
dc_uproc_message "Session Variable File: $_SESSION_XML_FILE created successfully"
unset _SESSION_XML_FILE _SESSION_VARS _V _VV _TD
return 0
}
```

Example Session Variables

```
<?xml version="1.0"?>
<!-- Company:ASDEVL Area:S Session:DP_SEPFTPM -->
<!-- SessionVersion:001 SessionNo:0001458 -->
<!-- CreatedByUPROC: S_ASDEVL_PROCS:DP_HEP-FTP.000 At:2009-11-16 15:50:26
   -->
<defines>
  <session>
    <DP_SOURCE_FILE>/home/adpc/cservap/at/du_test/my_f*.txt</DP_SOURCE_FILE>
    <EPRINT_USERID>jason_bourne</EPRINT_USERID>
    <EPRINT_PSWD>very_secret</EPRINT_PSWD>
    <DP_FTP_HOST>chara.cc.umanitoba.ca</DP_FTP_HOST>
  </session>
</defines>
```

Retrieving Session Variables

```
dc_uproc_get_session_variables() {
if  [ -z "$S_CODSESS" ]; then
    dc_uproc_message "dc_uproc_get_session_variables can only be used in sessions"
    exit 16
fi
_SESSION_XML_FILE="${U_TMP_PATH}/${S_ESPEXE}_${S_CODSESS}_${S_NUMSESS}.xml"
if  [ ! -r "$_SESSION_XML_FILE" ]; then
    dc_uproc_message "dc_uproc_get_session_variables $_SESSION_XML_FILE not found"
    exit 16
fi
_VARS=`dc_generate_parms -r session -f "$_SESSION_XML_FILE"`
if  [ $? -eq 0 ]; then
    if  [ -n "$_VARS" ]; then
        eval "$_VARS"
    else
        dc_uproc_message "dc_uproc_get_session_variables - no variables returned"
        exit 16
    fi
else
    dc_uproc_message "dc_uproc_get_session_variables - variable setting error"
    exit 16
fi
unset _VARS _SESSION_XML_FILE
return 0
}
```

Dereference XML

```
dc_generate_parms() {
LANGUAGE="sh"
PARMSFILE="unknown"
OPTIND=1
while getopts r:l:f: FLAG; do
    case $FLAG in
        r) ENVIRONMENT=$OPTARG;;
        l) LANGUAGE=$OPTARG;;
        f) PARMSFILE=$OPTARG;;
        *) echo "Usage: $0 -r environment -l lang -f parmsfile"
           return 16;;
    esac
done
XSLFILE="$XSLDIR/${LANGUAGE}.xslt"
if  [ -r "$PARMSFILE" ]; then
    echo "$0 - the parms file $PARMSFILE is not readable"
    return 16
fi
xsltproc --stringparam e "$ENVIRONMENT" $XSLFILE $PARMSFILE
RC=$?
return $RC
}
```

XSLT for Shell Scripts

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:output method="text"/>
  <xsl:param name="e"/>
  <xsl:template match="/">
    <xsl:for-each select="defines/*">
      <xsl:if test="name(.) = $e">
        <xsl:for-each select="./*">
          <xsl:value-of select="name(.)"/>
          <xsl:text>="</xsl:text>
          <xsl:value-of select="."/>
          <xsl:text>";</xsl:text>
          <xsl:text>
</xsl:text>
        </xsl:for-each>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

3. End of UPROC Cleanup

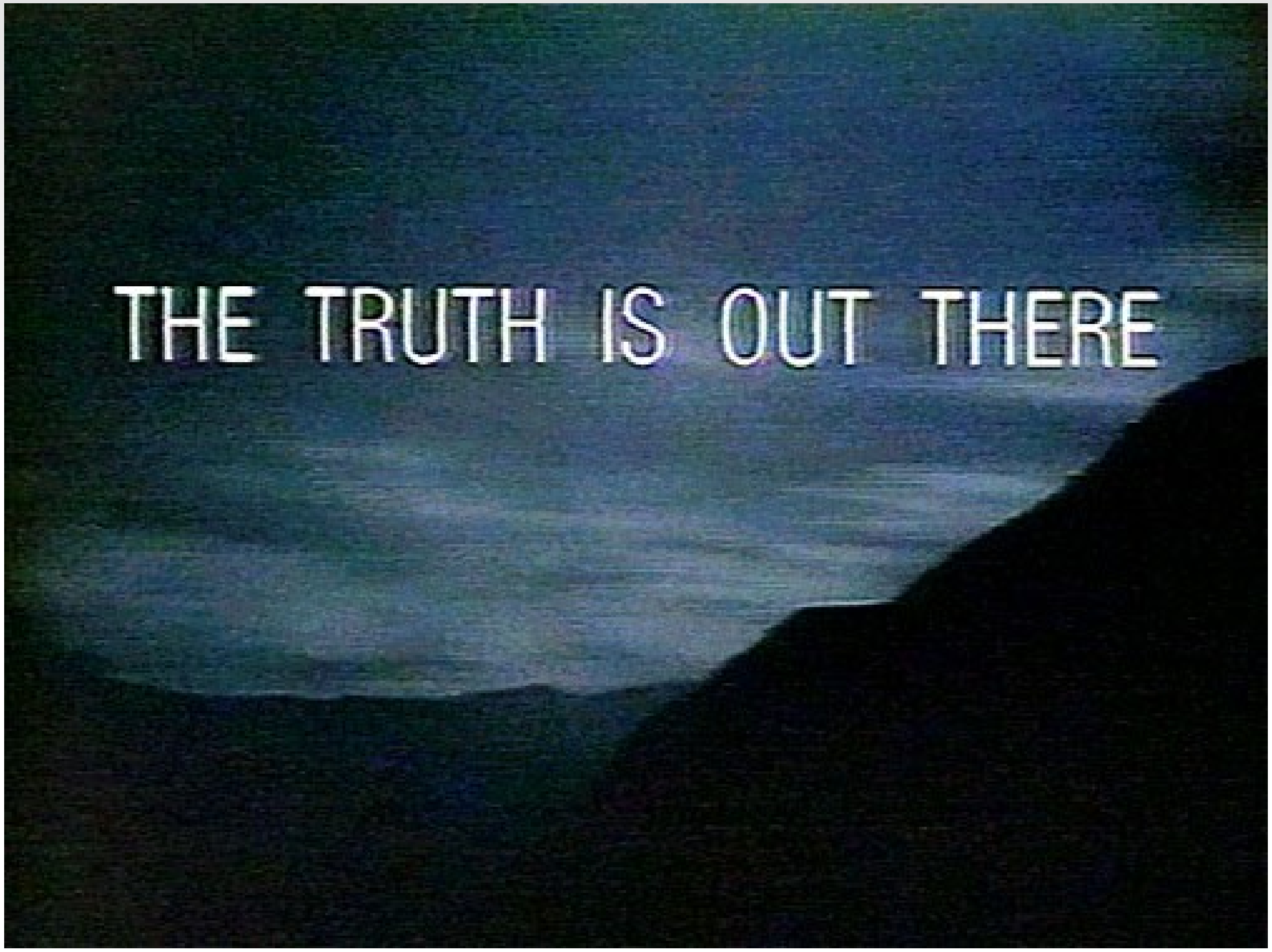
```
#------------------------------------------------------------------------
# Function..........: dc_uproc_end
# Author ...........: Gary F. Alderson
# Date-Written......: 2008-06-23
# Purpose...........: To perform any end of job cleanup for for every
#                     uproc.
#------------------------------------------------------------------------
dc_uproc_end() {
    _RC=$?
    [ $_RC -gt $DC_MAXCC ] && DC_MAXCC=$_RC
    dc_uproc_message "--------------------------------------------------"
    dc_uproc_message "End of uproc processing"
    dc_uproc_message "--------------------------------------------------"
    if  [ $DC_MAXCC -ge $DC_FAILURE_LIMIT ]; then
        dc_uproc_message "RC $DC_MAXCC exceeds limit $DC_FAILURE_LIMIT"
        dc_uproc_message "UPROC is considered to have failed."
        DC_RESEXE=$DC_MAXCC
    else
        DC_RESEXE=0
    fi
    return $DC_RESEXE
}
```

3. End of UPROC Cleanup

For Windows folks ...

```
@echo off
rem dc_uproc_end.bat
if  not defined UXEXE goto noenvset
rem ---------------------------------------
rem --- Perform end of UPROC processing ---
rem ---------------------------------------
call dc_uproc_message "End of uproc processing"
goto end
:noenvset
echo The $Universe environment has not been set.
set DC_MAXCC=%DC_FAILURE_LIMIT%
:end
set RESEXE=0
if  %DC_MAXCC% GEQ %DC_FAILURE_LIMIT% set RESEXE=%DC_MAXCC%
```

Questions and Hopefully Answers



Thank You

- Thank you ORSYP for holding this event today.
- Thank you all for participating.
- This presentation will be available to you soon.
- Contact Information:
  - Email: Gary.Alderson@Aldersoft.CA
  - LinkedIn: http://ca.linkedin.com/in/GaryFAlderson
  - Twitter: @GaryFAlderson