Rocket

U2 Support's U2 Replication Monitoring Phantoms and Exception Scripts

Deployment and Usage

March 2014

Last Revision 29th March 2019

Notices

Edition

Publication date: March 2014
Product Versions: UniVerse 11.1,11.2, 11.3 and 12.1 UniData 7.2, 7.3 8.1 and 8.2

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Contact information

Website: www.rocketsoftware.com
Rocket Software, Inc. Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA
Tel: +1 781 577 4321
Fax: +1 617 630 7100

Contacting Global Technical Support

If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Customer Portal to report and track a problem, to submit an enhancement request or question, or to find answers in the U2 Knowledgebase. The Rocket Customer Portal is the primary method of obtaining support.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

If you do not already have a Rocket Customer Portal account, you can request one by clicking **Need an Account?** on the Rocket Customer Portal login page.

Alternatively, you can contact Global Technical Support by email or by telephone:

Email: u2support@rocketsoftware.com
Telephone:

North America	+1 800 729 3553
United Kingdom/France	+44 (0) 800 773 771 or +44 (0) 20 8867 3691
Europe/Africa	+44 (0) 20 8867 3692
Australia	+1 800 707 703 or +61 (0) 29412 5450
New Zealand	+0800 505 515

Contents

Chapter 1: Introduction to U2 Replication Monitoring and Exception Handling

The purpose of this document is to demonstrate and make available the approaches and tools that the U2 Team has taken to assist customers both internal and external in successful deployments of U2 Replication.

U2 Replication is a very powerful, very configurable but very complex part of both UniData and UniVerse. Given its complexity our current recommendation is that U2 Replication should not be deployed without the assistance of Rocket Solutioning Services. See the Disaster Recovery Notice.

Many customers have historically become very useful in deploying there U2 Applications into environments where the people responsible for those environments have little or no specific U2 Administration knowledge. This has been one of the many positives of deploying U2 over a long period time and we'd like that to continue.

To administer and maintain U2 Replication then specific U2 Administration skills and knowledge needs to be acquired. To minimize this requirement, the processes and tools that we discuss in this document are designed to act as proactive notifications of any problems, which may have otherwise gone unnoticed and hence become a large problem later.

We also discuss how the use of an exception script when correctly designed can be used to automatically reestablish replication after some things as network interruptions.

Exception Action Script

U2 Replication contains a configurable exception action in the repsys file. This configurable is called EXCEPTION_ACTION. On the UNIX implementations this script is UNIX shell script and on the Windows implementation, it is a bat file. In the configurable you define the absolute path to the script name.

The EXCEPTION_ACTION trigger works in two ways within U2 Replication and provides an administrator the ability a slot to run their own code in the event of any non-DBA ordered suspension of replication. (I.e. if replication was suspended because of a request entered from the command prompt or via the XAdmin tool then the trigger is not activated).

There is one other event within U2 Replication that will trigger the EXCEPTION_ACTION and that is if the inbuilt 'heartbeat' encounters a TIMEOUT. U2 Replication has an inbuilt 'heartbeat' that is in addition to any O/S level timeouts applied to the TCP/IP communication layers. Each replication group sends a control packet between the publisher and subscriber approximately every 3 seconds. If these control packets go unacknowledged for the number of seconds defined by the TIMEOUT configurable or if the TCP/IP communication layer is lost then Replication is suspended and the EXCEPTION_ACTION is triggered.

Over a period of time we've developed and fine-tuned a three-phase exception action script. The first phase reports any unexpected suspension and can be used to send an email notification of the suspension to a predefined list of recipients. The second and third phases of the script are only run on the subscribing system as if they were also to run on the publisher this could result in a reciprocal firing of the script.

The second phase of the script then makes sure that all the replication groups have been successfully suspended and again it can notify the predefined list of recipients with the results of that check. The third and final phase only proceeds if the second phase has confirmed a successful suspension of all the groups.

The third phase of the script attempts to reestablish replication between the machines via the use of replication sync command. Again, the predefined list of recipients can be notified of the results of this action.

The examples provided use mailx/sendmail in UNIX as the email client and on windows it's a smtp Gmail account. As with all the examples and programs provided here they are 'as is'. Support will not be assisting in their modification or the setup of any required mail client that's needed within your environment to ensure their correct operation. If further assistance is required then please refer to the following notice DisaterRecoveryNotice.

UNIX Exception Script

The example exception scripts for UNIX are distributed within the phantom monitor deployment program which is discussed later in this document (See the InstallButton in the VB deployment program section). Both the UniData and UniVerse versions also support the passing in of two arguments.

- '-x' will turn on standard script debugging using the 'set –x' command
- '-v' will get the script just to report the script version then exit

Testing the script

The script is designed to be run via the firing of the EXCEPTION_ACTION phrase being activated within UniData / UniVerse. It relies on several environment variables that are only set if the script is fired off in the expected manner.

Attempting to run the script outside of this environment is not recommended and is not a valid test of the script. If you need advice / assistance on testing the script please discuss with Rocket Support or Rocket Services who can assist you.

Emails Produced within the script

The shell script variable MAILRECS defines the list of email recipients who should receive any email notification sent from the script in relation to the firing of the script and the details of the steps performed within the script. In conjunction with the MAILRECS variable the SENDMAILCMD variable defines which command should be used to send the email notifications.

The example in the script is based around the standard UNIX sendmail command. If you wish to use a different mail command then the areas of the script where the sendmail command is used will need modification for your installation.

The script will send several emails and constructs the sendmail command as follows: -

```
echo "Subject: REPLICATION DISABLED" | cat - $reportfilename |
$SENDMAILCMD -F $machinename -t $MAILRECS
```

The script should be reviewed for all occurrences of sendmail and modified as required.

Environment/Shell Variables

The script uses several Environment and Shell variables during its running. Some of these can be changed to meet your installation requirements and others shouldn't be changed. The following table describes the variables that can be changed and explains some of the other major reference variables used in the script. If the variable is not described in the table below then it would be unwise to change it unless you understand shell script and are confident of the change you are making.

Variables	Changeable	Usage / Description
MAILREC	Y	Discussed in previous section
SENDMAILCMD	Y	Discussed in previous section
UDTHOME / UVHOME	Y	Used to determine the install location of UV/UD. Default is /apps/ud or /app/uv
UDTBIN/UVBIN	Y	Used with UDTHOME/UVHOME default is UDTHOME/bin
PUB_STOP PUB_EXIT PUB_SHUTDOWN PUB_RUNNIG PUB_DO_RECONFIG PUB_DO_BACKSYNC PUB_DO_FAILOVER REP_RUNNING REP_SUSPENDED REP_SYNCING REP_DO_SYNC REP_EXIT REP_DISABLED SUB_STOP SUB_EXIT SUB_SHUTDOWN SUB_RUNNING SUB_DO_RECONFIG SUB_SYNCING SUB_DO_FAILOVER SUB_RESYNCING	N	Hardcoded to match the return codes produced from the ud/uv_repadmin report command. The script uses these to determine the status of replication.
REP_EXCEP_SCRIPT_VERSION	Y	Used to mark version of scripts
CROSS_GROUP_TRANSACTION	Y	This defaults to 0 and when set to 0 will check to see if Cross Group Transactions are active on the machine. If you set this to 1 it means the script will always assume Cross Group Transactions are present and will avoid the overhead of checking. It is recommended you set to this to 1 if you know your installation uses Cross Group Transactions
PUBCHECKMIN SUBCHECKMIN	Y	These values default to 2 and are used to ensure that once the script has completed the script will

DISCHECKMIN		not be run for another 2 minutes even if requested to. This is to avoid the script being fired off in repeated succession.
reportfilename	Y	Defines the location that the trace file from the script is placed
scriptrunningname	Y	Defines the name of the control file that the script uses to ensure that only one instance of the script can be running at any one time
RESYNCSLEEP	Y	This values defaults to 120 and defines the number of seconds that the script will sleep on a subscriber before attempting to issue a uv/ud_repadmin sync command to correct the replication suspension.
publisher subscriber disabled unexpectedcode	N	Set by the script via the result of the ud/uv_repadmin report command in order to determine the status of the system on which the script is running
LOGKEEPCNT	Y	Determines the number of previous versions of the trace file will be kept the default is 5
scriptruncountfile	Y	Defines the name of the file that contains the last run counter

Walkthrough of the Script Logic

1. The environment variables UDTHOME / UVHOME / UDTBIN / UVBIN are checked and if not set prior to the script execution are set to the defaults in the script.
2. The ud_reapdmin status code environment variables are set, along with the script version and cross group transaction override check variables
3. The script checks to see if a debug request or version request was passed to it via an ‘x’ or ‘v’ option.
    a. If a version request is made the version is reported and the script exits
    b. If a debug request is made the debug option is set for the script
4. The scriptrunningname variable is checked
    a. If the file is present, the presence of the file is reported to the log file and an email notification sent and the script exists
5. The lastruncount is determined from the scriptruncountfile and copies the last tracefile to tracefile.lastruncount
    a. The lastruncount is incremented by 1, if this exceeds the maximum count then the lastruncount is set to 1 and written back to the scriptruncountfile
6. The ud/uv_repadmin report command is used to determine the system status within replication. The result can be

    a. A Publisher
    b. A Subscriber
    c. Disabled
    d. Undetermined
7. The next allowed run time of the script is calculated
    a. If the system time is before that time, the script exits
8. An email is sent out from the script detailing that replication was suspended along with the contents of the script trace log
    a. If the system is not a subscriber the script exits at this point as the recovery attempt will only be run from a subscriber to avoid unwanted reciprocal firing of the script
9. Each group is checked in turn to ensure it has been correctly suspended. Each group will be checked up to 10 times with a 5 second sleep between each check cycle.
    a. An email notification of the success / failure of the checking process is then sent out
    b. If the script was unable to confirm the suspension of all the groups an email is sent to this effect with the script trace and the script will exit at this point.
10. The script then sleeps for the defined RESYNCSLEEP period
11. The CROSS_GROUP_TRANSACTION variable is then checked
    a. If not set the script will use the reptool/uvreptool engineering interface to determine if cross group transactions are present on the system
12. If the CROSS_GROUP_TRANSACTION variable is set or cross group transactions are present on the system then one single ud/uv_repadmin sync command is sent otherwise a separate sync command is sent to each group in turn.
    a. Ideally, we should be able to sync each group in turn but due to an unresolved product issue if cross group transactions are present sending single sync commands will result in a stalled condition in replication.
13. The success or failure of the sync command(s) are then emailed out along with the script trace and the script exits at this point.

Notes

Step 4 and Step 6 are designed to ensure that should multiple suspension events be encountered in quick succession that the script only runs once during those events. This is based on customer experience that if multiple copies of the script are running at the same time can result in further problems in such 'panic' situations.

Step 7 ensures that any auto recovery attempt only takes place on a non-disabled subscribing system. This is again based on customer experience in that allowing both systems to attempt recovery can also result in further complications and a possible reciprocal firing of the script on the other system. In terms of disablement If replication is disabled a 'sync' command will fail.

UniData Example

```
#!/bin/sh
#
# Replication Status Code that returned by ud_repadmin report command
#
# (c) Rocket Software 2014 - 2019 All Rights Reserved
# Disclaimer of Warranties. Rocket Software disclaims to the fullest extent
authorized by law any and all other warranties, whether express or implied,
including, without limitation, any implied warranties
# of  merchantability or fitness for a particular purpose. Without limitation
of the foregoing, Rocket Software expressly does not warrant that:
# (a) the software will meet your requirements [or expectations];
# (b) the software or the software content] will be free of bugs, errors,
viruses or other defects;
# (c) any results, output, or data provided through or generated by the
software will be accurate, up-to-date, complete or reliable;
# (d) the software will be compatible with third party software;
# (e) any errors in the software will be corrected.
#
#   1.3.1 JDS 25/02/14 First Proposed Customer Release
#   1.3.2 JDS 20/05/15 Corrected check to be /.uvhome not .uvhome
#   1.3.5 JDS 03/06/15 Change to all_synced check for disablement
#   1.4.7 JDS 22/01/16 Small Spelling Correction
#   1.4.8 JDS 27/01/16 Change to ignore publishing group errors on Recovery
#                      This is for sites with mixed publishing and
subscribing groups
#                      On the same machine
#   1.5.0 JDS 16/11/16 Add configurable for CGT (Cross Group Transactions)
and also try to detect CGT
#                      If CGT is detected or set Script will sync all groups
at once to
#                      avoid Replication Stall of one by one sync and CGT
#   1.5.1 JDS 20/12/16 Modify CGT sync call to remove DISTRIB phrase
#   1.5.2 JDS 22/12/16 Correct version number and correct $filenane typo to
$filename
#                      and deal with a blank file for last run time
#   1.5.3 JDS 27/03/17 Email notification if it is found that script is
already running
#   1.5.4 JDS 15/09/17 Move sendmail command to an environment variable to
allow modification in one place
#   1.5.5 JDS 29/11/17 Ensure running file is removed before exit 2 and
expand pub / sub check
#   1.5.8 JDS 29/03/18 Added second showud / showuv after suspension testing
```

```
#                       Added abilty to keep previous outputs of the execption
log file
#   1.5.9 JDS 29/05/18 Moved Declaration of machinename to avoid a sender
being set to '-t' when script is
#                       already running
#
# Define Mail receipents and sendmail cmd
export MAILRECS="someone@you.com,someoneelse@you.com"
export SENDMAILCMD="/usr/lib/sendmail"
# Check UDTHOME
udthome="$UDTHOME"
if [ "$udthome" = "" ]
then
  export UDTHOME=/apps/ud
fi
udtbin="$UDTBIN"
if [ "$udtbin" = "" ]
then
  export UDTBIN=$UDTHOME/bin
fi
# Publishing group Status Codes
export PUB_STOP=1 #/* group is normally stopped */
export PUB_EXIT=2 #/* group has quit abnormally.*/
export PUB_SHUTDOWN=3 #/* group is doing shutdown */
export PUB_RUNNING=4 #/* group is in running mode */
export PUB_DO_RECONFIG=5 #/* group is doing reconfigure */
export PUB_DO_BACKSYNC=6 #/* group is doing backsync */
export PUB_DO_FAILOVER=7 #/* group is doing failover */
# Publishing Replication Status Codes
export REP_RUNNING=11 #/* replication is in running mode */
export REP_SUSPENDED=12 #/* replication is in suspension mode */
export REP_SYNCING=13 #/* replication is in syncing mode */
export REP_DO_SUSPEND=14 #/* replication is doing suspend */
export REP_DO_SYNC=15 #/* replication is doing sync */
export REP_EXIT=16 #/* replication exit */
export REP_DISABLED=17 #/* replication disabled */
# Subscribing Replication/group status codes.
export SUB_STOP=21 #/* group is normally stopped */
export SUB_EXIT=22 #/* group has quit abnormally.*/
export SUB_SHUTDOWN=23 #/* group is doing shutdown */
export SUB_RUNNING=24 #/* replication is in running mode */
export SUB_DO_RECONFIG=25 #/* group is doing reconfigure */
export SUB_DO_SUSPEND=27 #/* replication is suspending */
export SUB_SYNCING=28 #/* replication is doing sync */
export SUB_DO_FAILOVER=29 #/* group is doing failover */
export SUB_RESYNCING=30 #/* group is doing re-syncing */
# Script Version
export REP_EXCEP_SCRIPT_VERSION=1.5.9
export CROSS_GROUP_TRANSACTION=0
export debugrequest=0
```

```
export versionrequest=0
while getopts ":xv" opt; do
case $opt in
x ) debugrequest=1 ;;
v ) versionrequest=1 ;;
esac
done
if [ $debugrequest = 1 ]
then
  set -x
fi
if [ $versionrequest = 1 ]
then
 echo "Script Version is :"$REP_EXCEP_SCRIPT_VERSION
 exit 0
fi
# Number of minutes to check before running script again
export PUBCHECKMIN=2
export SUBCHECKMIN=2
export DISCHECKMIN=2
# Sleep is in seconds not minutes
export RESYNCSLEEP=120
# Set number of Result logs to keep
export LOGKEEPCNT=5
# Show exception time
export dstring=`date`
export reportfilename=$UDTHOME'/RepExcept.errlog'
export scriptrunningname=$UDTHOME'/RepExcept.running'
export scriptruncountfile=$UDTHOME/RepExcept.cnt
export machinename=`uname -a | awk '{print $2}'`
if [ -f $scriptrunningname ]
then
  echo $scriptrunningname" found script already running - exiting"
  echo "Subject: Exception Script is Already Running - New Request Rejected"
| cat - $reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
  exit 5
fi
if [ -f $scriptruncountfile ]
then
  export lastruncount=`head -1 $scriptruncountfile | tail -1`
  lastruncount=`expr $lastruncount + 1`
  if [ $lastruncount -gt $LOGKEEPCNT ]
  then
    lastruncount=1
  fi
  echo $lastruncount > $scriptruncountfile
  export cpcmd="cp $reportfilename $reportfilename$lastruncount"
  $cpcmd
else
  echo "0" > $scriptruncountfile
```

```
fi
echo "Run Started at :"$dstring > $scriptrunningname
echo "UniData Replication Exception called at:"$dstring > $reportfilename
echo "Script Version is:"$REP_EXCEP_SCRIPT_VERSION >> $reportfilename
echo "Run on machine "$machinename >> $reportfilename
echo "Is this the publisher or subscriber" >> $reportfilename
export cmd="$UDTBIN/ud_repadmin report $UNIREP_REMOTESYS"
echo "Now running:"$cmd >> $reportfilename
$cmd
returncode=$?
echo "returncode=$returncode" >> $reportfilename
export publisher=0
export subscriber=0
export disabled=0
export unexpectedcode=0
if [ $returncode -lt 17 ]
then
  publisher=1
fi
if [ $returncode -gt 17 ]
then
  subscriber=1
fi
if [ $returncode -gt 30 ]
then
  unexpectedcode=1
fi
if [ $returncode = "17" ]
then
  disabled=1
fi
if [ $publisher = 1 ]
then
  echo "System is a publisher" >> $reportfilename
fi
if [ $subscriber = 1 ]
then
  echo "System is a subscriber" >> $reportfilename
fi
if [ $disabled = 1 ]
then
  echo "Replication is Disabled !" >> $reportfilename
fi
if [ $unexpectedcode = 1 ]
then
  echo "Returncode is outside expected Range, so the assumption Machine is a
subscriber may be incorrect, hence will not perform recovery " >>
$reportfilename
fi
# Check Last Run Time and avoid running multiple times
```

```
export checkfilename=$UDTHOME'/RepExcept.runtime'
export ddatetime=`date +"%Y %j %H %M"`
ddatetime=`echo $ddatetime | sed 's/ //g'`
export run_check=0
if [ ! -f $checkfilename ]
then
  # Put the current date time into the log file
  echo $ddatetime > $checkfilename
  run_check=1
else
  ldatetime=`cat $checkfilename`
  if [ "$ldatetime" = "" ]
  then
    echo "Empty Last Run File Found" >> $reportfilename
    echo $ddatetime > $checkfilename
    run_check=1
  else
    if [ $publisher = 1 ]
    then
      tdatetime=`expr $ldatetime + $PUBCHECKMIN`
    fi
    if [ $subscriber = 1 ]
    then
      tdatetime=`expr $ldatetime + $SUBCHECKMIN`
    fi
    if [ $disabled = 1 ]
    then
      tdatetime=`expr $ldatetime + $DISCHECKMIN`
    fi
    echo "Last run at "$ldatetime >> $reportfilename
    echo "Current "$ddatetime >> $reportfilename
    echo "Next "$tdatetime >> $reportfilename
    if [ $ddatetime -gt $tdatetime ]
    then
      run_check=1
      echo $ddatetime > $checkfilename
    fi
  fi
fi
echo "Run Check "$run_check >> $reportfilename
if [ $run_check = 0 ]
then
  rm $scriptrunningname
  exit
fi
# Show environment variables passed from replication system
echo "UNIREP_REMOTESYS="$UNIREP_REMOTESYS >> $reportfilename
echo "UNIREP_REPTYPE="$UNIREP_REPTYPE >> $reportfilename
echo "UNIREP_GRPNAME="$UNIREP_GRPNAME >> $reportfilename
echo "UNIREP_ERRCODE="$UNIREP_ERRCODE >> $reportfilename
```

```
echo "UNIREP_ERRSTRING="$UNIREP_ERRSTRING >> $reportfilename
echo "UDTBIN=$UDTBIN" >> $reportfilename
echo "Return Error Codes" >> $reportfilename
echo "NO ERROR=0" >> $reportfilename
echo "PUB_STOP="$PUB_STOP >> $reportfilename
echo "PUB_EXIT="$PUB_EXIT >> $reportfilename
echo "PUB_SHUTDOWN="$PUB_SHUTDOWN >> $reportfilename
echo "PUB_RUNNING="$PUB_RUNNING >> $reportfilename
echo "PUB_DO_RECONFIG="$PUB_DO_RECONFIG >> $reportfilename
echo "PUB_DO_BACKSYNC ="$PUB_DO_BACKSYNC >> $reportfilename
echo "PUB_DO_FAILOVER="$PUB_DO_FAILOVER >> $reportfilename
echo "REP_RUNNING="$REP_RUNNING >> $reportfilename
echo "REP_SUSPENDED="$REP_SUSPENDED >> $reportfilename
echo "REP_SYNCING="$REP_SYNCING >> $reportfilename
echo "REP_DO_SUSPEND="$REP_DO_SUSPEND >> $reportfilename
echo "REP_DO_SYNC="$REP_DO_SYNC >> $reportfilename
echo "REP_EXIT="$REP_EXIT >> $reportfilename
echo "REP_DISABLED="$REP_DISABLED >> $reportfilename
echo "SUB_STOP="$SUB_STOP >> $reportfilename
echo "SUB_EXIT="$SUB_EXIT >> $reportfilename
echo "SUB_SHUTDOWN="$SUB_SHUTDOWN >> $reportfilename
echo "SUB_RUNNING="$SUB_RUNNING >> $reportfilename
echo "SUB_DO_RECONFIG="$SUB_DO_RECONFIG >> $reportfilename
echo "SUB_DO_SUSPEND="$SUB_DO_SUSPEND >> $reportfilename
echo "SUB_SYNCING="$SUB_SYNCING >> $reportfilename
echo "SUB_DO_FAILOVER="$SUB_DO_FAILOVER >> $reportfilename
echo "SUB_RESYNCING="$SUB_RESYNCING >> $reportfilename
# Run ud_repadmin report command
export cmd="$UDTBIN/ud_repadmin report -detail $UNIREP_REMOTESYS"
echo "Now running:"$cmd >> $reportfilename
$cmd >> $reportfilename
echo "showud" >> $reportfilename
$UDTBIN/showud >> $reportfilename
if [ $publisher = 1 ]
then
  echo "System Reported it was a publisher no need to try sync" >>
$reportfilename
fi
if [ $disabled = 1 ]
then
  echo "System Reported Replication was disabled no need to try sync" >>
$reportfilename
fi
if [ $unexpectedcode = 1 ]
then
  echo "Unexpected returncode was fouund so sync will not be attempted" >>
$reportfilename
fi
echo "Please See "$reportfilename" for more information" >> $reportfilename
# SendMail of Failure
```

```
echo "Attempting First SendMail" >> $reportfilename
if [ $disabled = 1 ]
then
   echo "Subject: REPLICATION DISABLED" | cat - $reportfilename |
$SENDMAILCMD -F $machinename -t $MAILRECS
   echo "Replication Disabled Email Sent" >> $reportfilename
else
   echo "Subject: REPLICATION SUSPENDED" | cat - $reportfilename |
$SENDMAILCMD -F $machinename -t $MAILRECS
   echo "Replication Suspended Email Sent" >> $reportfilename
fi
if [ $publisher = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
if [ $disabled = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
if [ $unexpectedcode = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
# Check all groups are suspended before proceeding
# Not Needed for Publisher or if Replication is disabled
export all_suspended=0
export try=0
while [ $all_suspended = 0 -a $try -lt 10 ]
do
  sleep 5
  all_suspended=1
  menu_str="2\n0\n0\n0\n\n\n"
  for grpname in `printf $menu_str | $UDTBIN/reptool | grep "^Group" | awk
'{print $3}'`; do
     if [ $all_suspended ]
     then
       # Run ud_repadmin report command
     export cmd="$UDTBIN/ud_repadmin REPORT GROUP $grpname DISTRIB
$UNIREP_REMOTESYS"
     echo "Try Counter is "$try >> $reportfilename
     echo "Now running:"$cmd >> $reportfilename
     $cmd
     returncode=$?
     echo "returncode=$returncode" >> $reportfilename
     if [ $returncode -gt 30 ]
     then
 # Report command failed
```

```
        echo "Report command failed, exit." >> $reportfilename
            rm $scriptrunningname
        exit 2
     elif [ $returncode = $REP_SUSPENDED -o $returncode = $SUB_STOP -o
$returncode = $SUB_EXIT ]
     then
        echo "Group $grpname is suspended." >> $ filename
     else
        all_suspended=0
     fi
  fi
  done
  try=`expr $try + 1`
done
echo "showud" >> $reportfilename
$UDTBIN/showud >> $reportfilename
if [ $all_suspended = 0 ]
then
   echo "Number of Try's Exceeded to Confirm All Groups Suspended" >>
$reportfilename
   echo "Subject: UNABLE TO CONFIRM SUSPENSION OF ALL GROUPS (NO SYNC WILL BE
ATTEMPTED)" | cat - $reportfilename | $SENDMAILCMD -F $machinename -t
$MAILRECS
   echo "Unable to confirm suspension email sent" >> $reportfilename
   rm $scriptrunningname
   exit 3
fi
echo "Subject: CONFIRMED SUSPENSION OF ALL GROUPS - Resync Attempt in 2
Minutes" | cat - $reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
echo "Confirmed suspension email sent" >> $reportfilename
# Try automatic resync and resume - only run on subscriber
echo "Sleeping for $RESYNCSLEEP" >> $reportfilename
sleep $RESYNCSLEEP
echo "Woke up from Sleep" >> $reportfilename

echo "Cross Group Transaction Currently set to $CROSS_GROUP_TRANSACTION" >>
$reportfilename
if [ $CROSS_GROUP_TRANSACTION = 0 ]
then
  echo "Testing for Transactions" >> $reportfilename
  menu_str="6\n1\n0\n0\n\n\n"
  trans_cnt=`printf $menu_str | $UDTBIN/reptool | grep "^TCR Created:" | awk
'{print $3}'`
  if [ $trans_cnt = 0 ]
  then
    echo "No Transactions Seen Leaving CGT Unset" >> $reportfilename
  else
    echo "Transactions Found so setting CGT" >> $reportfilename
      CROSS_GROUP_TRANSACTION=1
  fi
```

```
fi

export all_synced=1
export group_syncing=1
if [ $CROSS_GROUP_TRANSACTION = 0 ]
then
menu_str="2\n0\n0\n0\n\n\n"
for grp_name in `printf $menu_str | $UDTBIN/reptool | grep "^Group" | awk
'{print $3}'`; do
   cmd="$UDTBIN/ud_repadmin sync -wait -verbose GROUP $grp_name DISTRIB
$UNIREP_REMOTESYS"
   echo "Now running:"$cmd >> $reportfilename
   $cmd >> $reportfilename
   returncode=$?
   echo "Return code=$returncode" >> $reportfilename
   if [ $returncode = 0 ]
   then
      echo "Group Sync Worked" >> $reportfilename
   else
      export cmd="$UDTBIN/ud_repadmin report GROUP $grp_name"
      echo "Now running:"$cmd >> $reportfilename
      $cmd
      returncode=$?
      echo "returncode=$returncode" >> $reportfilename
      if [ $returncode = $REP_SYNCING -o $returncode = $REP_DO_SYNC -o
$returncode = $REP_RUNNING  -o $returncode = $SUB_RUNNING  -o $returncode =
$SUB_SYNCING -o $returncode = $SUB_RESYNCING ]
      then
         group_syncing=1
      else
           group_syncing=0
      fi
        if [ $returncode -lt 17 ]
        then
           group_syncing=1
             echo "group appears to be a publishing group so and sync
failures are ignored"
        fi
      echo "group_syncing=$group_syncing" >> $reportfilename
      if [ $group_syncing = 0 ]
      then
         echo "Group Sync Failed" >> $reportfilename
         all_synced=0
      fi
   fi
   sleep 1
done
else
   cmd="$UDTBIN/ud_repadmin sync -wait -verbose $UNIREP_REMOTESYS"
   echo "Now running:"$cmd >> $reportfilename
```

```
   $cmd >> $reportfilename
   returncode=$?
   echo "Return code=$returncode" >> $reportfilename
   if [ $returncode = 0 ]
   then
      echo "Whole Sync Worked" >> $reportfilename
   else
      export cmd="$UDTBIN/ud_repadmin report"
      echo "Now running:"$cmd >> $reportfilename
      $cmd
      returncode=$?
      echo "returncode=$returncode" >> $reportfilename
      if [ $returncode = $REP_SYNCING -o $returncode = $REP_DO_SYNC -o
$returncode = $REP_RUNNING  -o $returncode = $SUB_RUNNING  -o $returncode =
$SUB_SYNCING -o $returncode = $SUB_RESYNCING ]
      then
         group_syncing=1
      else
           group_syncing=0
      fi
        if [ $returncode -lt 17 ]
        then
           group_syncing=1
             echo "group appears to be a publishing group so and sync
failures are ignored"
        fi
      echo "group_syncing=$group_syncing" >> $reportfilename
      if [ $group_syncing = 0 ]
      then
         echo "Whole Sync Failed" >> $reportfilename
         all_synced=0
      fi
   fi
   sleep 1
fi

echo "all_synced flag = "$all_synced >> $reportfilename
if [ $all_synced = 1 ]
then
 echo "Subject: Sync Commmands Worked - Please Check Systems" | cat -
$reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
else
 echo "Subject: SYNC COMMANDS FAILED - PLEASE CHECK SYSTEMS" | cat -
$reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
 rm $scriptrunningname
 exit 4
fi
rm $scriptrunningname
exit 0
```

UniVerse Example

```
#!/bin/sh
#
# Replication Status Code that returned by uv_repadmin report command
#
# (c) Rocket Software 2014 - 2019 All Rights Reserved
# Disclaimer of Warranties. Rocket Software disclaims to the fullest extent
authorized by law any and all other warranties, whether express or implied,
including, without limitation, any implied warranties
# of  merchantability or fitness for a particular purpose. Without limitation
of the foregoing, Rocket Software expressly does not warrant that:
# (a) the software will meet your requirements [or expectations];
# (b) the software or the software content] will be free of bugs, errors,
viruses or other defects;
# (c) any results, output, or data provided through or generated by the
software will be accurate, up-to-date, complete or reliable;
# (d) the software will be compatible with third party software;
# (e) any errors in the software will be corrected.
#
#   1.3.1 JDS 25/02/14 First Proposed Customer Release
#   1.3.2 JDS 20/05/15 Corrected check to be /.uvhome not .uvhome
#   1.3.5 JDS 03/06/15 Change to all_synced check for disablement
#   1.4.7 JDS 16/01/16 Spelling Change and Corrected Check for Sync Failure
Subject in Email
#   1.4.8 JDS 27/01/16 Change to ignore publishing group errors on Recovery
#                      This is for sites with mixed publishing and
subscribing groups
#                      On the same machine
#   1.5.0 JDS 16/11/16 Add configurable for CGT (Cross Group Transactions)
and also try to detect CGT
#                      If CGT is detected or set Script will sync all groups
at once to
#                      avoid Replication Stall of one by one sync and CGT
#   1.5.1 JDS 20/12/16 Modify CGT sync call to remove DISTRIB phrase
#   1.5.2 JDS 22/12/16 Correct Version number and correct $filenane typo to
$filename
#                      and deal with a blank file for last run time
#   1.5.3 JDS 27/03/17 Email notification if it is found that script is
already running
```

```
#   1.5.4 JDS 15/09/17 Move sendmail command to an environment variable to
allow modification in one place
#   1.5.5 JDS 29/11/17 Ensure running file is removed before exit 2 and
expand pub / sub check
#   1.5.8 JDS 29/03/18 Added second showud / showuv after suspension testing
#                      Added abilty to keep previous outputs of the execption
log file
#   1.5.9 JDS 29/05/18 Moved Declaration of machinename to avoid a sender
being set to '-t' when script is
#                      already running
#
# Define Mail receipents and sendmail cmd
export MAILRECS="someone@you.com,someoneelse@you.com"
export SENDMAILCMD="/usr/lib/sendmail"
# Check UVHOME
uvhome="$UVHOME"
if [ "$uvhome" = "" ]
then
  export UVHOME=`cat /.uvhome`
fi
uvbin="$UVBIN"
if [ "$uvbin" = "" ]
then
  export UVBIN=$UVHOME/bin
fi
# Publishing group Status Codes
export PUB_STOP=1 #/* group is normally stopped */
export PUB_EXIT=2 #/* group has quit abnormally.*/
export PUB_SHUTDOWN=3 #/* group is doing shutdown */
export PUB_RUNNING=4 #/* group is in running mode */
export PUB_DO_RECONFIG=5 #/* group is doing reconfigure */
export PUB_DO_BACKSYNC=6 #/* group is doing backsync */
export PUB_DO_FAILOVER=7 #/* group is doing failover */
# Publishing Replication Status Codes
export REP_RUNNING=11 #/* replication is in running mode */
export REP_SUSPENDED=12 #/* replication is in suspension mode */
export REP_SYNCING=13 #/* replication is in syncing mode */
export REP_DO_SUSPEND=14 #/* replication is doing suspend */
export REP_DO_SYNC=15 #/* replication is doing sync */
export REP_EXIT=16 #/* replication exit */
export REP_DISABLED=17 #/* replication disabled */
# Subscribing Replication/group status codes.
export SUB_STOP=21 #/* group is normally stopped */
```

```
export SUB_EXIT=22 #/* group has quit abnormally.*/
export SUB_SHUTDOWN=23 #/* group is doing shutdown */
export SUB_RUNNING=24 #/* replication is in running mode */
export SUB_DO_RECONFIG=25 #/* group is doing reconfigure */
export SUB_DO_SUSPEND=27 #/* replication is suspending */
export SUB_SYNCING=28 #/* replication is doing sync */
export SUB_DO_FAILOVER=29 #/* group is doing failover */
export SUB_RESYNCING=30 #/* group is doing re-syncing */
# Script Version
export REP_EXCEP_SCRIPT_VERSION=1.5.9
export CROSS_GROUP_TRANSACTION=0
export debugrequest=0
export versionrequest=0
while getopts ":xv" opt; do
case $opt in
x ) debugrequest=1 ;;
v ) versionrequest=1 ;;
esac
done
if [ $debugrequest = 1 ]
then
  set -x
fi
if [ $versionrequest = 1 ]
then
 echo "Script Version is :"$REP_EXCEP_SCRIPT_VERSION
 exit 0
fi
# Number of minutes to check before running script again
export PUBCHECKMIN=2
export SUBCHECKMIN=2
export DISCHECKMIN=2
# Sleep is in seconds not minutes
export RESYNCSLEEP=120
# Set number of Result logs to keep
export LOGKEEPCNT=5
# Show exception time
export dstring=`date`
export reportfilename=$UVHOME'/RepExcept.errlog'
export scriptrunningname=$UVHOME'/RepExcept.running'
export scriptruncountfile=$UVHOME/RepExcept.cnt
export machinename=`uname -a | awk '{print $2}'`
if [ -f $scriptrunningname ]
```

```
then
  echo $scriptrunningname" found script already running - exiting"
  echo "Subject: Exception Script is Already Running - New Request Rejected"
| cat - $reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
  exit 5
fi
if [ -f $scriptruncountfile ]
then
  export lastruncount=`head -1 $scriptruncountfile | tail -1`
  lastruncount=`expr $lastruncount + 1`
  if [ $lastruncount -gt $LOGKEEPCNT ]
  then
    lastruncount=1
  fi
  echo $lastruncount > $scriptruncountfile
  export cpcmd="cp $reportfilename $reportfilename$lastruncount"
  $cpcmd
else
  echo "0" > $scriptruncountfile
fi
echo "Run Started at :"$dstring > $scriptrunningname
echo "UniVerse Replication Exception called at:"$dstring > $reportfilename
echo "Script Version is:"$REP_EXCEP_SCRIPT_VERSION >> $reportfilename
echo "Run on machine "$machinename >> $reportfilename
echo "Is this the publisher or subscriber" >> $reportfilename
export cmd="$UVBIN/uv_repadmin report $UNIREP_REMOTESYS"
echo "Now running:"$cmd >> $reportfilename
$cmd
returncode=$?
echo "returncode=$returncode" >> $reportfilename
export publisher=0
export subscriber=0
export disabled=0
export unexpectedcode=0
if [ $returncode -lt 17 ]
then
  publisher=1
fi
if [ $returncode -gt 17 ]
then
  subscriber=1
fi
if [ $returncode -gt 30 ]
```

```
then
  unexpectedcode=1
fi
if [ $returncode = "17" ]
then
  disabled=1
fi
if [ $publisher = 1 ]
then
  echo "System is a publisher" >> $reportfilename
fi
if [ $subscriber = 1 ]
then
  echo "System is a subscriber" >> $reportfilename
fi
if [ $disabled = 1 ]
then
  echo "Replication is Disabled !" >> $reportfilename
fi
if [ $unexpectedcode = 1 ]
then
  echo "Returncode is outside expected Range, so the assumption Machine is a
subscriber may be incorrect, hence will not perform recovery " >>
$reportfilename
fi
# Check Last Run Time and avoid running multiple times
export checkfilename=$UVHOME'/RepExcept.runtime'
export ddatetime=`date +"%Y %j %H %M"`
ddatetime=`echo $ddatetime | sed 's/ //g'`
export run_check=0
if [ ! -f $checkfilename ]
then
  # Put the current date time into the log file
  echo $ddatetime > $checkfilename
  run_check=1
else
  ldatetime=`cat $checkfilename`
  if [ "$ldatetime" = "" ]
  then
    echo "Empty Last Run File Found" >> $reportfilename
    echo $ddatetime > $checkfilename
    run_check=1
  else
```

```
    if [ $publisher = 1 ]
    then
      tdatetime=`expr $ldatetime + $PUBCHECKMIN`
    fi
    if [ $subscriber = 1 ]
    then
      tdatetime=`expr $ldatetime + $SUBCHECKMIN`
    fi
    if [ $disabled = 1 ]
    then
      tdatetime=`expr $ldatetime + $DISCHECKMIN`
    fi
    echo "Last run at "$ldatetime >> $reportfilename
    echo "Current "$ddatetime >> $reportfilename
    echo "Next "$tdatetime >> $reportfilename
    if [ $ddatetime -gt $tdatetime ]
    then
      run_check=1
      echo $ddatetime > $checkfilename
    fi
  fi
fi
echo "Run Check "$run_check >> $reportfilename
if [ $run_check = 0 ]
then
  rm $scriptrunningname
  exit
fi
# Show environment variables passed from replication system
echo "UNIREP_REMOTESYS="$UNIREP_REMOTESYS >> $reportfilename
echo "UNIREP_REPTYPE="$UNIREP_REPTYPE >> $reportfilename
echo "UNIREP_GRPNAME="$UNIREP_GRPNAME >> $reportfilename
echo "UNIREP_ERRCODE="$UNIREP_ERRCODE >> $reportfilename
echo "UNIREP_ERRSTRING="$UNIREP_ERRSTRING >> $reportfilename
echo "UVBIN=$UVBIN" >> $reportfilename
echo "Return Error Codes" >> $reportfilename
echo "NO ERROR=0" >> $reportfilename
echo "PUB_STOP="$PUB_STOP >> $reportfilename
echo "PUB_EXIT="$PUB_EXIT >> $reportfilename
echo "PUB_SHUTDOWN="$PUB_SHUTDOWN >> $reportfilename
echo "PUB_RUNNING="$PUB_RUNNING >> $reportfilename
echo "PUB_DO_RECONFIG="$PUB_DO_RECONFIG >> $reportfilename
echo "PUB_DO_BACKSYNC ="$PUB_DO_BACKSYNC >> $reportfilename
```

```
echo "PUB_DO_FAILOVER="$PUB_DO_FAILOVER >> $reportfilename
echo "REP_RUNNING="$REP_RUNNING >> $reportfilename
echo "REP_SUSPENDED="$REP_SUSPENDED >> $reportfilename
echo "REP_SYNCING="$REP_SYNCING >> $reportfilename
echo "REP_DO_SUSPEND="$REP_DO_SUSPEND >> $reportfilename
echo "REP_DO_SYNC="$REP_DO_SYNC >> $reportfilename
echo "REP_EXIT="$REP_EXIT >> $reportfilename
echo "REP_DISABLED="$REP_DISABLED >> $reportfilename
echo "SUB_STOP="$SUB_STOP >> $reportfilename
echo "SUB_EXIT="$SUB_EXIT >> $reportfilename
echo "SUB_SHUTDOWN="$SUB_SHUTDOWN >> $reportfilename
echo "SUB_RUNNING="$SUB_RUNNING >> $reportfilename
echo "SUB_DO_RECONFIG="$SUB_DO_RECONFIG >> $reportfilename
echo "SUB_DO_SUSPEND="$SUB_DO_SUSPEND >> $reportfilename
echo "SUB_SYNCING="$SUB_SYNCING >> $reportfilename
echo "SUB_DO_FAILOVER="$SUB_DO_FAILOVER >> $reportfilename
echo "SUB_RESYNCING="$SUB_RESYNCING >> $reportfilename
# Run uv_repadmin report command
export cmd="$UVBIN/uv_repadmin report -detail $UNIREP_REMOTESYS"
echo "Now running:"$cmd >> $reportfilename
$cmd >> $reportfilename
echo "showuv" >> $reportfilename
$UVBIN/showuv >> $reportfilename
if [ $publisher = 1 ]
then
  echo "System Reported it was a publisher no need to try sync" >>
$reportfilename
fi
if [ $disabled = 1 ]
then
  echo "System Reported Replication was disabled no need to try sync" >>
$reportfilename
fi
if [ $unexpectedcode = 1 ]
then
  echo "Unexpected returncode was fouund so sync will not be attempted" >>
$reportfilename
fi
echo "Please See "$reportfilename" for more information" >> $reportfilename
# SendMail of Failure
echo "Attempting First SendMail" >> $reportfilename
if [ $disabled = 1 ]
then
```

```
   echo "Subject: REPLICATION DISABLED" | cat - $reportfilename |
$SENDMAILCMD -F $machinename -t $MAILRECS
   echo "Replication Disabled Email Sent" >> $reportfilename
else
   echo "Subject: REPLICATION SUSPENDED" | cat - $reportfilename |
$SENDMAILCMD -F $machinename -t $MAILRECS
   echo "Replication Suspended Email Sent" >> $reportfilename
fi
if [ $publisher = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
if [ $disabled = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
if [ $unexpectedcode = 1 ]
then
  rm $scriptrunningname
  exit 0
fi
# Check all groups are suspended before proceeding
# Not Needed for Publisher or if Replication is disabled
export all_suspended=0
export try=0
while [ $all_suspended = 0 -a $try -lt 10 ]
do
  sleep 5
  all_suspended=1
  menu_str="2\n0\n0\n0\n\n\n"
  for grpname in `printf $menu_str | $UVBIN/uvreptool | grep "^Group" | awk
'{print $3}'`; do
     if [ $all_suspended ]
     then
       # Run uv_repadmin report command
     export cmd="$UVBIN/uv_repadmin REPORT GROUP $grpname DISTRIB
$UNIREP_REMOTESYS"
     echo "Try Counter is "$try >> $reportfilename
     echo "Now running:"$cmd >> $reportfilename
     $cmd
     returncode=$?
```

```
     echo "returncode=$returncode" >> $reportfilename
     if [ $returncode -gt 30 ]
     then
 # Report command failed
        echo "Report command failed, exit." >> $reportfilename
            rm $scriptrunningname
        exit 2
     elif [ $returncode = $REP_SUSPENDED -o $returncode = $SUB_STOP -o
$returncode = $SUB_EXIT ]
     then
        echo "Group $grpname is suspended." >> $ filename
     else
        all_suspended=0
     fi
  fi
  done
  try=`expr $try + 1`
done
echo "showuv" >> $reportfilename
$UVBIN/showuv >> $reportfilename
if [ $all_suspended = 0 ]
then
   echo "Number of Try's Exceeded to Confirm All Groups Suspended" >>
$reportfilename
   echo "Subject: UNABLE TO CONFIRM SUSPENSION OF ALL GROUPS (NO SYNC WILL BE
ATTEMPTED)" | cat - $reportfilename | $SENDMAILCMD -F $machinename -t
$MAILRECS
   echo "Unable to confirm suspension email sent" >> $reportfilename
   rm $scriptrunningname
   exit 3
fi
echo "Subject: CONFIRMED SUSPENSION OF ALL GROUPS - Resync Attempt in 2
Minutes" | cat - $reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
echo "Confirmation suspension email sent" >> $reportfilename
# Try automatic resync and resume - only run on subscriber
echo "Sleeping for $RESYNCSLEEP" >> $reportfilename
sleep $RESYNCSLEEP
echo "Woke up from Sleep" >> $reportfilename

echo "Cross Group Transaction Currently set to $CROSS_GROUP_TRANSACTION" >>
$reportfilename
if [ $CROSS_GROUP_TRANSACTION = 0 ]
then
```

```
  echo "Testing for Transactions" >> $reportfilename
  menu_str="6\n1\n0\n0\n\n\n"
  trans_cnt=`printf $menu_str | $UVBIN/uvreptool | grep "^TCR Created:" | awk
'{print $3}'`
  if [ $trans_cnt = 0 ]
  then
    echo "No Transactions Seen Leaving CGT Unset" >> $reportfilename
  else
    echo "Transactions Found so setting CGT" >> $reportfilename
      CROSS_GROUP_TRANSACTION=1
  fi
fi

export all_synced=1
export group_syncing=1
if [ $CROSS_GROUP_TRANSACTION = 0 ]
then
menu_str="2\n0\n0\n0\n\n\n"
for grp_name in `printf $menu_str | $UVBIN/uvreptool | grep "^Group" | awk
'{print $3}'`; do
   cmd="$UVBIN/uv_repadmin sync -wait -verbose GROUP $grp_name DISTRIB
$UNIREP_REMOTESYS"
   echo "Now running:"$cmd >> $reportfilename
   $cmd >> $reportfilename
   returncode=$?
   echo "Return code=$returncode" >> $reportfilename
   if [ $returncode = 0 ]
   then
      echo "Group Sync Worked" >> $reportfilename
   else
      export cmd="$UVBIN/uv_repadmin report GROUP $grp_name"
      echo "Now running:"$cmd >> $reportfilename
      $cmd
      returncode=$?
      echo "returncode=$returncode" >> $reportfilename
      if [ $returncode = $REP_SYNCING -o $returncode = $REP_DO_SYNC -o
$returncode = $REP_RUNNING  -o $returncode = $SUB_RUNNING  -o $returncode =
$SUB_SYNCING -o $returncode = $SUB_RESYNCING ]
      then
         group_syncing=1
      else
           group_syncing=0
      fi
```

```
        if [ $returncode -lt 17 ]
        then
           group_syncing=1
             echo "group appears to be a publishing group so and sync
failures are ignored"
        fi
      echo "group_syncing=$group_syncing" >> $reportfilename
      if [ $group_syncing = 0 ]
      then
         echo "Group Sync Failed" >> $reportfilename
         all_synced=0
      fi
   fi
   sleep 1
done
else
   cmd="$UVBIN/uv_repadmin sync -wait -verbose $UNIREP_REMOTESYS"
   echo "Now running:"$cmd >> $reportfilename
   $cmd >> $reportfilename
   returncode=$?
   echo "Return code=$returncode" >> $reportfilename
   if [ $returncode = 0 ]
   then
      echo "Whole Sync Worked" >> $reportfilename
   else
      export cmd="$UVBIN/uv_repadmin report"
      echo "Now running:"$cmd >> $reportfilename
      $cmd
      returncode=$?
      echo "returncode=$returncode" >> $reportfilename
      if [ $returncode = $REP_SYNCING -o $returncode = $REP_DO_SYNC -o
$returncode = $REP_RUNNING  -o $returncode = $SUB_RUNNING  -o $returncode =
$SUB_SYNCING -o $returncode = $SUB_RESYNCING ]
      then
         group_syncing=1
      else
           group_syncing=0
      fi
        if [ $returncode -lt 17 ]
        then
           group_syncing=1
             echo "group appears to be a publishing group so and sync
failures are ignored"
```

```
        fi
      echo "group_syncing=$group_syncing" >> $reportfilename
      if [ $group_syncing = 0 ]
      then
         echo "Whole Sync Failed" >> $reportfilename
         all_synced=0
      fi
   fi
fi

echo "all_synced flag = "$all_synced >> $reportfilename
if [ $all_synced = 1 ]
then
 echo "Subject: Sync Commmands Worked - Please Check Systems" | cat -
$reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
else
 echo "Subject: SYNC COMMANDS FAILED - PLEASE CHECK SYSTEMS" | cat -
$reportfilename | $SENDMAILCMD -F $machinename -t $MAILRECS
 rm $scriptrunningname
 exit 4
fi
rm $scriptrunningname
exit 0
```

Windows Exception Script

To implement the same functionality into the Windows version of the exception script that was provided in the UNIX version the script is implemented as a Windows Powershell Script/

To run the program from a bat file as required in U2 Replication you need a bat file to allow execution of the powershell, our example is as follows

```
@ECHO OFF
powershell -ExecutionPolicy ByPass -File C:\u2\ud\RepWinExcep.ps1 -
windowstyle hidden
```

Testing the Windows Exception Script

The script has a '-v' option to display the version of the script along with some other key information.

```
C:\u2\ud>powershell -ExecutionPolicy ByPass -File C:\u2\ud\RepWinExcep.ps1 -v
UniData Home C:\u2\ud
UniData Bin C:\u2\ud\bin
UniData Version 8.2
Machine Name UK-L-JS03
Script Version 1.6.3
```

The script also has provision '-d' option for debugging to allow a test run of the script outside of the replication environment but does require the DebugRemoteSys variable to be set correctly.

```
C:\u2\ud>powershell -ExecutionPolicy ByPass -File C:\u2\ud\RepWinExcep.ps1 -d
Exception Action Powershell - Reached Normal End of Exception Action
Powershell
```

Walkthrough of the Script Logic

1. The environment variables UDTHOME / UVHOME / UDTBIN / UVBIN / U2VERSION are set from the Windows Registry Keys
2. The uv/ud_reapdmin status code environment variables are set, along with the script version and cross group transaction override check variables
3. The script checks to see if a debug request or version request was passed to it via an '-d' or -'v' option.

    a. If a version request is made the version is reported and the script exits
    b. If a debug request is made the debug option is set for the script
4. The scriptrunningname variable is checked
    a. If the file is present, the presence of the file is reported to the log file and an email notification sent and the script exists
5. The lastruncount is determined from the scriptruncountfile and copies the last tracefile to tracefile.lastruncount
    a. The lastruncount is incremented by 1, if this exceeds the maximum count then the lastruncount is set to 1 and written back to the scriptruncountfile
6. The ud/uv_repadmin report command is used to determine the system status within replication. The result can be
    a. A Publisher
    b. A Subscriber
    c. Disabled
    d. Undetermined
7. An email is sent out from the script detailing that replication was suspended along with the contents of the script trace log
    a. If the system is not a subscriber the script exits at this point as the recovery attempt will only be run from a subscriber to avoid unwanted reciprocal firing of the script
8. Each group is checked in turn to ensure it has been correctly suspended. Each group will be checked up to 10 times with a 5 second sleep between each check cycle.
    a. An email notification of the success / failure of the checking process is then sent out
    b. If the script was unable to confirm the suspension of all the groups an email is sent to this effect with the script trace and the script will exit at this point.
9. The script then sleeps for the defined RESYNCSLEEP period
10. The CROSS_GROUP_TRANSACTION variable is then checked
    a. If not set the script will use the reptool/uvreptool engineering interface to determine if cross group transactions are present on the system
11. If the CROSS_GROUP_TRANSACTION variable is set or cross group transactions are present on the system then one single ud/uv_repadmin sync command is sent otherwise a separate sync command is sent to each group in turn.
    a. Ideally, we should be able to sync each group in turn but due to an unresolved product issue if cross group transactions are present sending single sync commands will result in a stalled condition in replication.
12. The success or failure of the sync command(s) are then emailed out along with the script trace and the script exits at this point.

Notes

Step 4 is designed to ensure that should multiple suspension events be encountered in quick succession that the script only runs once during those events. This is based on customer experience

that if multiple copies of the script are running at the same time can result in further problems in such 'panic' situations.

Step 6 ensures that any auto recovery attempt only takes place on a non-disabled subscribing system. This is again based on customer experience in that allowing both systems to attempt recovery can also result in further complications and a possible reciprocal firing of the script on the other system. In terms of disablement If replication is disabled a 'sync' command will fail.

Exception Script Variables

Variables	Changeable	Usage / Description
EMailFrom	Y	The email address notifications will be sent from
EMailTo	Y	Who to send any email notifications to
SMTPServer	Y	The name of the SMTP mail server to send notifications from
SMTPPort	Y	The port of the SMTP mail server to connect to
SMTPSSL	Y	Boolean flag to determine if SMTP server requires SSL verification
SMTPUserName	Y	User Name on the SMTP server
SMTPPassword	Y	Password for the User Name above
DebugRemoteSys	Y	The name of the remote SYSTEM as described in the repsys file, so for a subscriber this will be the SYSTEM name of the publisher
UDTHOME / UVHOME	N	Pulled from the Windows Registry Keys
UDTBIN/UVBIN	N	Pulled from the Windows Registry Keys
U2VERSION	N	Pulled from the Windows Registry Keys
PUB_STOP PUB_EXIT PUB_SHUTDOWN PUB_RUNNIG PUB_DO_RECONFIG PUB_DO_BACKSYNC PUB_DO_FAILOVER REP_RUNNING REP_SUSPENDED REP_SYNCING REP_DO_SYNC REP_EXIT REP_DISABLED SUB_STOP SUB_EXIT SUB_SHUTDOWN SUB_RUNNING SUB_DO_RECONFIG SUB_SYNCING SUB_DO_FAILOVER SUB_RESYNCING	N	Hardcoded to match the return codes produced from the ud/uv_repadmin report command. The script uses these to determine the status of replication.

REP_EXCEP_SCRIPT_VERSION	Y	Used to mark version of scripts
CROSS_GROUP_TRANSACTION	Y	This defaults to 0 and when set to 0 will check to see if Cross Group Transactions are active on the machine. If you set this to 1 it means the script will always assume Cross Group Transactions are present and will avoid the overhead of checking. It is recommended you set to this to 1 if you know your installation uses Cross Group Transactions
PUBCHECKMIN SUBCHECKMIN DISCHECKMIN	Y	These values default to 2 and are used to ensure that once the script has completed the script will not be run for another 2 minutes even if requested to. This is to avoid the script being fired off in repeated succession.
reportfilename	Y	Defines the location that the trace file from the script is placed
scriptrunningname	Y	Defines the name of the control file that the script uses to ensure that only one instance of the script can be running at any one time
RESYNCSLEEP	Y	This values defaults to 120 and defines the number of seconds that the script will sleep on a subscriber before attempting to issue a uv/ud_repadmin sync command to correct the replication suspension.
LOGKEEPCNT	Y	Determines the number of previous versions of the trace file will be kept the default is 5
publisher subscriber disabled unexpectedcode	N	Set by the script via the result of the ud/uv_repadmin report command in order to determine the status of the system on which the script is running
scriptruncountfile	Y	Defines the name of the file that contains the last run counter
scriptworkinfile	Y	Scratch work file name
scriptworkoutfile	Y	Scratch work file name
scriptworkgrouplist	Y	Scratch work file name
machinename	N	Hostname of the machine running the script

UniData and UniVerse RepWinExcepVariables.psm1

Both the UniVerse and UniData windows powershell examples use a psm1 file where some of the variables you may wish to change such as the details around using Email and the default remote system name.

```
class Variables
{
    static [string] $EmailFrom = "sender@example.com"
    static [string] $EmailTo = "recip1@example.com,recip2@example.com"
    static [string] $SMTPServer = "smtp.gmail.com"
    static [string] $SMTPPort = 587
    static [string] $SMTPSSL = $true
    static [string] $SMTPUserName = "xxxxxxxxxxxx"
    static [string] $SMTPPassword = "xxxxxxxxxxxx"
    static [string] $DebugRemoteSys = "primary"

}
```

UniData Example

```
#
# Example Replication Exception PowerShell File for UniData
#
# (c) Rocket Software 2014 - 2019 All Rights Reserved
# Disclaimer of Warranties. Rocket Software disclaims to the fullest extent
authorized by law any and all other warranties, whether express or implied,
including, without limitation, any implied warranties
# of  merchantability or fitness for a particular purpose. Without limitation
of the foregoing, Rocket Software expressly does not warrant that:
# (a) the software will meet your requiments [or expectations];
# (b) the software or the software content] will be free of bugs, errors,
viruses or other defects;
# (c) any results, output, or data provided through or generated by the
software will be accurate, up-to-date, complete or reliable;
# (d) the software will be compatible with third party software;
# (e) any errors in the software will be corrected.
#
#  1.6.0 JDS 07/02/19 First Windows Version
#  1.6.1 JDS 27/02/19 Changes made after Customer beta testing SRL @ CACI
#  1.6.3 JDS 29/03/19 First Customer Release Version
#
Using module ".\RepWinExcepVariables.psm1"
#
# Check for -v (Version Switch) or -d (Debug Switch)
Param(
      [switch][bool]$v,
      [switch][bool]$d
)
# Get Registry Values for UniData
$KEY = "HKLM:\SOFTWARE\Rocket Software\UniData\CurrentVersion"
$ErrorActionPreference = "stop"
Try {
      $U2VERSION = (Get-ItemProperty $KEY).UDVersion
}
Catch {
      Write-Output "Exception Action Powershell - Unable to find Current
UniData Version in the Registry"
      Exit 1
}
$KEY = "HKLM:\SOFTWARE\Rocket Software\UniData\" + $U2VERSION
Try {
      $UDTHOME = (Get-ItemProperty $KEY).UDTHOME
}
Catch {
      Write-Output "Exception Action Powershell - Unable to find UniData Home
in the Registry"
      Exit 2
```

```
}
Try {
      $UDTBIN = (Get-ItemProperty $KEY).UDTBIN
}
Catch {
      Write-Output "Exception Action Powershell - Unable to find UniData Bin
in the Registry"
      Exit 3
}
# Publishing group Status Codes
$PUB_STOP = 1
$PUB_EXIT = 2
$PUB_SHUTDOWN = 3
$PUB_RUNNING = 4
$PUB_DO_RECONFIG = 5
$PUB_DO_BACKSYNC = 6
$PUB_DO_FAILOVER = 7
# Publishing Replication Status Codes
$REP_RUNNING = 11
$REP_SUSPENDED = 12
$REP_SYNCING = 13
$REP_DO_SUSPEND = 14
$REP_DO_SYNC = 15
$REP_EXIT = 16
$REP_DISABLED = 17
# Subscribing Replication/group status codes.
$SUB_STOP = 21
$SUB_EXIT = 22
$SUB_SHUTDOWN = 23
$SUB_RUNNING = 24
$SUB_DO_RECONFIG = 25
$SUB_DO_SUSPEND = 27
$SUB_SYNCING = 28
$SUB_DO_FAILOVER = 29
$SUB_RESYNCING = 30
# Script Version
$REP_EXCEP_SCRIPT_VERSION = "1.6.3"
# Number of minutes to check before running script again
$PUBCHECKMIN = 2
$SUBCHECKMIN = 2
$DISCHECKMIN = 2
# Sleep is in seconds not minutes
$RESYNCSLEEP = 120
# Set number of Result logs to keep
$LOGKEEPCNT = 5
# Cross Group Transactions Flag
$global:CROSS_GROUP_TRANSACTION = $false
# Set Exception Log File Names
$reportfilename = $UDTHOME + "\RepExcept.errlog"
$scriptrunningname = $UDTHOME + "\RepExcept.running"
```

```
$scriptruncountfile = $UDTHOME + "\RepExcept.cnt"
$scriptworkinfile = $UDTHOME + "\RepExcept.inwrk"
$scriptworkoutfile = $UDTHOME + "\RepExcept.outwrk"
$scriptworkgrouplist = $UDTHOME + "\RepExcept.grplist"
# Result Variables set in Functions
$global:allsuspended = $true
$global:allsynced = $true
# Get Machine Name
$machinename = [string](hostname)
# Check -v switch
if ($v.IsPresent) {
      Write-Output "UniData Home $UDTHOME"
      Write-Output "UniData Bin $UDTBIN"
      Write-Output "UniData Version $U2VERSION"
      Write-Output "Machine Name $machinename"
      Write-Output "Script Version $REP_EXCEP_SCRIPT_VERSION"
      exit 0
}
# Declare Send-Email Function
function Send-Email {
      $Body = ""
      $regex = ""
      foreach ($line in Get-Content $reportfilename) {
            if ($line -match $regex) {
                  $Body = $Body + $line + "`r`n"
            }
      }
      "To: " + [Variables]::EmailTo >> $reportfilename
      "Subject: " + $Subject >> $reportfilename
      try {
            $SMTPClient = New-Object
Net.Mail.SmtpClient([Variables]::SMTPServer, [Variables]::SMTPPort)
            $SMTPClient.EnableSsl = [Variables]::SMTPSSL
            $SMTPClient.Credentials = New-Object
System.Net.NetworkCredential([Variables]::SMTPUserName,
[Variables]::SMTPPassword);
            $SMTPClient.Send([Variables]::EmailFrom, [Variables]::EmailTo,
$Subject, $Body)
      }
      catch {
            "Failed to send email" >> $reportfilename
            $_.Exception.GetType().FullName + " " + $_.Exception.Message >>
$reportfilename
      }
      "" >> $reportfilename
}
# Declare Function to remove running file check
function Remove-Run-File {
      try {
            Remove-Item -Path $scriptrunningname
```

```
      }
      catch {
            Write-Output "Exception Action Powershell - Unable to remove
$scriptrunningname"
            $Subject = "U2Replication $machinename - Unable to remove
$scriptrunningname"
            Send-Email
            exit 6
      }
}
# Declare Function to get group names
function Get-Group-Names {
      "Running ud_repadmin report -detail $UNIREP_REMOTESYS to get list of
groups" >> $reportfilename
      ud_repadmin report -detail $UNIREP_REMOTESYS > $scriptworkoutfile
      "" >> $reportfilename
      "Group Names" >> $reportfilename
      $grpcnt = 0
      foreach ($line in Get-Content $scriptworkoutfile) {
            if ($line -match $regex) {
                  $tline = $line.Trim()
                  if ($tline -ne "") {
                        $words = $tline.Split(" ")
                        if (($words[0] -eq "GROUP") -and ($words[2] -eq
"DISTRIBUTION")) {
                              $groupname = $words[1].Substring(0,
$words[1].Length - 1)
                              $groupname >> $reportfilename
                              $grpcnt++
                              if ($grpcnt -eq 1) {
                                    $groupname > $scriptworkgrouplist
                     } else {
                                    $groupname >> $scriptworkgrouplist
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
}
# Declare Function to check that each group is suspended
function Check-All-Suspended {
      "Checking to see if all the groups are Suspended" >> $reportfilename
      $trystart = 1
      foreach ($line in Get-Content $scriptworkgrouplist) {
            if ($line -match $regex) {
                  $groupname = $line
                  for ($trycnt = $trystart; $trycnt -le 10; $trycnt++) {
                        "Running ud_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS" >> $reportfilename
```

```
                        ud_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS
                        "Return Code was $LASTEXITCODE" >> $reportfilename
                        if ($LASTEXITCODE -gt $SUB_RESYNCING) {
                              "Report command failed, exiting powershell" >>
$reportfilename
                              Write-Output "Exception Action Powershell -
ud_repadmin encountered an error and terminated"
                              $Subject = "U2Replication $machinename -
ud_repadmin report encountered an error"
                              Send-Email
                              Remove-Run-File
                              exit 7
                  } else {
                              if (($LASTEXITCODE -eq $REP_SUSPENDED) -or
($LASTEXITCODE -eq $SUB_STOP) -or ($LASTEXITCODE -eq $SUB_EXIT)) {
                                    "Group is suspended" >> $reportfilename
                                    $trycnt = 11
                     } else {
                                    if ($trycnt -eq 10) {
                                          $global:allsuspended = $false
                                          "Could not confirm Suspension of
group, number of retry attempts exceeded" >> $reportfilename
                           } else {
                                          "Could not confirm Suspension of
group, retry attempt $trycnt, sleep 5 seconds and retry" >> $reportfilename
                                          Start-Sleep -Seconds 5
                                          $trystart += 1
                                    }
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
      showud >> $reportfilename
      "" >> $reportfilename
}
# Declare Function to perform sync attempt to recover suspension
function Do-Sync {
      # Check for Transactions as if Transactions are present then we cannot
sync group by group until bug is fixed that stops it from working
      if (-Not($global:CROSS_GROUP_TRANSACTION)) {
            "Checking to see Transactions are being used as indicator not
set" >> $reportfilename
            6 > $scriptworkinfile
            1 >> $scriptworkinfile
            0 >> $scriptworkinfile
            0 >> $scriptworkinfile
            Get-Content $scriptworkinfile | reptool > $scriptworkoutfile
```

```
            foreach ($line in Get-Content $scriptworkoutfile) {
                  if ($line -match $regex) {
                        $tline = $line.Trim()
                        if ($tline -ne "") {
                              $words = $tline.Split(" ")
                              if (($words[0] -eq "TCR") -and ($words[1] -eq
"Created:")) {
                                    $numtcr = $words[2]
                                    if (-Not($numtcr -eq "0")) {
                                          $global:CROSS_GROUP_TRANSACTION =
$true
                                          "Transactions detected" >>
$reportfilename
                                    }
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
      if ($global:CROSS_GROUP_TRANSACTION) {
            "Running ud_repadmin sync -wait -verbose $UNIREP_REMOTESYS" >>
$reportfilename
            ud_repadmin sync -wait -verbose $UNIREP_REMOTESYS >>
$reportfilename
   } else {
            foreach ($line in Get-Content $scriptworkgrouplist) {
                  $groupname = $line
                  "Running ud_repadmin sync -wait -verbose GROUP $groupname
DISTRIB $UNIREP_REMOTESYS" >> $reportfilename
                  ud_repadmin sync -wait -verbose GROUP $groupname DISTRIB
$UNIREP_REMOTESYS >> $reportfilename
            }
      }
      # Check to see if all groups were sync'd okay
      foreach ($line in Get-Content $scriptworkgrouplist) {
            $groupname = $line
            "Running ud_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS" >> $reportfilename
            ud_repadmin report GROUP $groupname DISTRIB $UNIREP_REMOTESYS >>
$reportfilename
            "Return Code was $LASTEXITCODE" >> $reportfilename
            if (($LASTEXITCODE -eq $REP_SYNCING) -or ($LASTEXITCODE -eq
$REP_DO_SYNC) -or ($LASTEXITCODE -eq $REP_RUNNING) -or ($LASTEXITCODE -eq
$SUB_RUNNING) -or ($LASTEXITCODE -eq $SUB_SYNCING) -or ($LASTEXITCODE -eq
$SUB_RESYNCING) -or ($LASTEXITCODE -lt $REP_DISABLED)) {
                  $groupsyncing = $true
                  "Group is Syncing or Sync'd" >> $reportfilename
        } else {
                  $global:allsynced = $false
```

```
                  "Group is not Syncing or Sync'd" >> $reportfilename
            }
      }
      "" >> $reportfilename
}
Function Copy-Last-Run-Output {
      Try {
            $lastruncount = Get-Content $scriptruncountfile
      }
      Catch {
            $lastruncount = 0
            $lastruncount > $scriptruncountfile
      }
      $lastruncount = [int]($lastruncount)
      $lastruncount++
      $nextruncount = $lastruncount
      if ($nextruncount -gt $LOGKEEPCNT) {
            $nextruncount = 1
      }
      $nextruncount > $scriptruncountfile
      $copyname = $reportfilename + [string]$nextruncount
      Try {
            Copy-Item -Path $reportfilename -Destination $copyname
      }
      Catch {
            Write-Output "Exception Action Powershell - Unable to copy
$reportfilename to $copyname"
      }
}
# Check to see if Script is Already Running
if (Test-Path -Path $scriptrunningname) {
      Write-Output "Exception Action Powershell - Found $scriptrunningname -
This indicates the exception script is currently running"
      $Subject = "U2Replication $machinename - Found $scriptrunningname -
This indicates the exception script is currently running"
      Send-Email
      exit 4
}
# Check to see if script is running from Replication Exception and get
Replication Environment Variables
if ($d.IsPresent) {
      # If Debug Flag is in place - bypass Replication Environment Check
      $UNIREP_REMOTESYS = [Variables]::DebugRemoteSys
      $UNIREP_REPTYPE = ""
      $UNIREP_GRPNAME = ""
      $UNIREP_ERRCODE = ""
      $UNIREP_ERRSTRING = ""
} else {
      Try {
            $UNIREP_REMOTESYS = Get-Content Env:\UNIREP_REMOTESYS
```

```
	}
	Catch {
		Write-Output "Exception Action Powershell - not being run from
Replication Exception"
		exit 5
	}
	Try {
		$UNIREP_REPTYPE = Get-Content Env:\UNIREP_REPTYPE
	}
	Catch {
		Write-Output "Exception Action Powershell - not being run from
Replication Exception"
		exit 5
	}
	Try {
		$UNIREP_GRPNAME = Get-Content Env:\UNIREP_GRPNAME
	}
	Catch {
		Write-Output "Exception Action Powershell - not being run from
Replication Exception"
		exit 5
	}
	Try {
		$UNIREP_ERRCODE = Get-Content Env:\UNIREP_ERRCODE
	}
	Catch {
		Write-Output "Exception Action Powershell - not being run from
Replication Exception"
		exit 5
	}
	Try {
		$UNIREP_ERRSTRING = Get-Content Env:\UNIREP_ERRSTRING
	}
	Catch {
		Write-Output "Exception Action Powershell - not being run from
Replication Exception"
		exit 5
	}
}
Try {
	$dstring = Get-Date
	# Create Scriptrunning File
	"Run Started at $dstring" > $scriptrunningname
	# If a previous report file exists make a copy of it
	if (Test-Path -Path $reportfilename) {
		Copy-Last-Run-Output
	}
	# Start new report file
	"UniData Replication Exception called at $dstring" > $reportfilename
	"Script Version=$REP_EXCEP_SCRIPT_VERSION" >> $reportfilename
```

```
      "UniData Home=$UDTHOME" >> $reportfilename
      "UniData Bin=$UDTBIN" >> $reportfilename
      "UniData Version=$U2VERSION" >> $reportfilename
      "Machine Name=$machinename" >> $reportfilename
      "" >> $reportfilename
      "Replication Environment Variables" >> $reportfilename
      "UNIREP_REMOTESYS=$UNIREP_REMOTESYS" >> $reportfilename
      "UNIREP_REPTYPE=$UNIREP_REPTYPE" >> $reportfilename
      "UNIREP_GRPNAME=$UNIREP_GRPNAME" >> $reportfilename
      "UNIREP_ERRCODE=$UNIREP_ERRCODE" >> $reportfilename
      "UNIREP_ERRSTRING=$UNIREP_ERRSTRING" >> $reportfilename
      "" >> $reportfilename
      "Replication Status Error Codes" >> $reportfilename
      "PUB_STOP=$PUB_STOP" >> $reportfilename
      "PUB_EXIT=$PUB_EXIT" >> $reportfilename
      "PUB_SHUTDOWN=$PUB_SHUTDOWN" >> $reportfilename
      "PUB_RUNNING=$PUB_RUNNING" >> $reportfilename
      "PUB_DO_RECONFIG=$PUB_DO_RECONFIG" >> $reportfilename
      "PUB_DO_BACKSYNC=$PUB_DO_BACKSYNC" >> $reportfilename
      "PUB_DO_FAILOVER=$PUB_DO_FAILOVER" >> $reportfilename
      "REP_RUNNING=$REP_RUNNING" >> $reportfilename
      "REP_SUSPENDED=$REP_SUSPENDED" >> $reportfilename
      "REP_SYNCING=$REP_SYNCING" >> $reportfilename
      "REP_DO_SUSPEND=$REP_DO_SUSPEND" >> $reportfilename
      "REP_DO_SYNC=$REP_DO_SYNC" >> $reportfilename
      "REP_EXIT=$REP_EXIT" >> $reportfilename
      "REP_DISABLED=$REP_DISABLED" >> $reportfilename
      "SUB_STOP=$SUB_STOP" >> $reportfilename
      "SUB_EXIT=$SUB_EXIT" >> $reportfilename
      "SUB_SHUTDOWN=$SUB_SHUTDOWN" >> $reportfilename
      "SUB_RUNNING=$SUB_RUNNING" >> $reportfilename
      "SUB_DO_RECONFIG=$SUB_DO_RECONFIG" >> $reportfilename
      "SUB_DO_SUSPEND=$SUB_DO_SUSPEND" >> $reportfilename
      "SUB_SYNCING=$SUB_SYNCING" >> $reportfilename
      "SUB_DO_FAILOVER=$SUB_DO_FAILOVER" >> $reportfilename
      "SUB_RESYNCING=$SUB_RESYNCING" >> $reportfilename
      "" >> $reportfilename
      cd $UDTBIN
      showud >> $reportfilename
      "" >> $reportfilename
      $publisher = $false
      $subscriber = $false
      $disabled = $false
      $unexpectedcode = $false
      "Running ud_repadmin report -detail $UNIREP_REMOTESYS" >>
$reportfilename
      ud_repadmin report -detail $UNIREP_REMOTESYS >> $reportfilename
      "" >> $reportfilename
      "Is this system a publisher or subscriber or is replication disabled ?
Exit Code was $LASTEXITCODE" >> $reportfilename
```

```
      if ($LASTEXITCODE -lt $REP_DISABLED) {
            $publisher = $true
            "System is a publisher - No Recovery will take place" >>
$reportfilename
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Publisher)"
      }
      if ($LASTEXITCODE -gt $REP_DISABLED) {
            $subscriber = $true
            "System is a subscriber" >> $reportfilename
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Subscriber)"
      }
      if ($LASTEXITCODE -eq $REP_DISABLED) {
            $disabled = $true
            "Replication is disabled" >> $reportfilename
            $Subject = "U2Replication $machinename - REPLICATION DISABLED"
      }
      if ($LASTEXITCODE -gt $SUB_RESYNCING) {
            $unexpectedcode = $true
            $subscriber = $false
            "Return code is outside expected Range, so the assumption the
System is a subscriber may be incorrect - No recovery will be performed" >>
$reportfilename
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Subscriber with Unexpected Code)"
      }
      "" >> $reportfilename
      "Sending Email Notification of Suspension" >> $reportfilename
      "" >> $reportfilename
      Send-Email
      # Only Run Recovery Steps if System is a subscriber
      if ($subscriber) {
            Get-Group-Names
            Check-All-Suspended
            if ($global:allsuspended) {
                  Write-Output "Exception Action Powershell - All groups
suspended - Powershell Sleep for $RESYNCSLEEP seconds"
                  $Subject = "U2Replication $machinename - Confirmed
Suspension of the all Groups - Resync in 2 Minutes (Subscriber)"
                  "Sleeping for $RESYNCSLEEP seconds before sync attempt" >>
$reportfilename
                  Send-Email
                  Start-Sleep -Seconds $RESYNCSLEEP
                  "Woken up from Sleep" >> $reportfilename
                  "" >> $reportfilename
                  Do-Sync
                  If ($global:allsynced) {
                        Write-Output "Exception Action Powershell - Confirmed
Sync of the all Groups"
```

```
                        $Subject = "U2Replication $machinename - Confirmed
Sync of the all Groups (Subscriber)"
        } else {
                        Write-Output "Exception Action Powershell - Unable to
Confirm Sync of the all Groups"
                        $Subject = "U2Replication $machinename - Unable to
Confirm Sync of the all Groups (Subscriber)"
                  }
                  Send-Email
   } else {
                  Write-Output "Exception Action Powershell - Unable to
confirm all groups suspended"
                  $Subject = "U2Replication $machinename - Unable to Confirm
Suspension of the all Groups - No Resync (Subscriber)"
                  Send-Email
            }
      }
}
Catch {
      Remove-Run-File
      Write-Output $_.Exception|format-list -force
      Write-Output "Exception Action Powershell - An Error Occurred"
      exit 8
}
Remove-Run-File
Write-Output "Exception Action Powershell - Reached Normal End of Exception
Action Powershell"
exit 0
```

UniVerse Example

```
#
# Example Replication Exception PowerShell File for UniVerse
#
# (c) Rocket Software 2014 - 2019 All Rights Reserved
# Disclaimer of Warranties. Rocket Software disclaims to the fullest extent
authorized by law any and all other warranties, whether express or implied,
including, without limitation, any implied warranties
# of  merchantability or fitness for a particular purpose. Without limitation
of the foregoing, Rocket Software expressly does not warrant that:
# (a) the software will meet your requirments [or expectations];
# (b) the software or the software content] will be free of bugs, errors,
viruses or other defects;
# (c) any results, output, or data provided through or generated by the
software will be accurate, up-to-date, complete or reliable;
# (d) the software will be compatible with third party software;
# (e) any errors in the software will be corrected.
#
#  1.6.0 JDS 07/02/19 First Windows Version
#  1.6.1 JDS 27/02/19 Changes made after Customer beta testing feedback SRL @
CACI
#  1.6.3 JDS 29/03/19 First Customer Release Version
#
Using module ".\RepWinExcepVariables.psm1"
#
# Check for -v (Version Switch) or -d (Debug Switch)
Param(
      [switch][bool]$v,
      [switch][bool]$d
)
# Get Registry Values for UniVerse
$KEY="HKLM:\SOFTWARE\Rocket Software\UniVerse\CurrentVersion"

$ErrorActionPreference = "stop"
Try {
      $U2VERSION = (Get-ItemProperty $KEY).UvVersion
}
Catch {
      Write-Output "Exception Action Powershell - Unable to find Current
UniVerse Version in the Registry"
      Exit 1
}
$KEY = "HKLM:\SOFTWARE\Rocket Software\UniVerse\" + $U2VERSION
Try {
         $UVHOME=(Get-ItemProperty $KEY).UvHOME
}
Catch {
```

```
      Write-Output "Exception Action Powershell - Unable to find UniVerse
Home in the Registry"
      Exit 2
}
$UVBIN=$UVHOME + "\bin"
# Publishing group Status Codes
$PUB_STOP = 1
$PUB_EXIT = 2
$PUB_SHUTDOWN = 3
$PUB_RUNNING = 4
$PUB_DO_RECONFIG = 5
$PUB_DO_BACKSYNC = 6
$PUB_DO_FAILOVER = 7
# Publishing Replication Status Codes
$REP_RUNNING = 11
$REP_SUSPENDED = 12
$REP_SYNCING = 13
$REP_DO_SUSPEND = 14
$REP_DO_SYNC = 15
$REP_EXIT = 16
$REP_DISABLED = 17
# Subscribing Replication/group status codes.
$SUB_STOP = 21
$SUB_EXIT = 22
$SUB_SHUTDOWN = 23
$SUB_RUNNING = 24
$SUB_DO_RECONFIG = 25
$SUB_DO_SUSPEND = 27
$SUB_SYNCING = 28
$SUB_DO_FAILOVER = 29
$SUB_RESYNCING = 30
# Script Version
$REP_EXCEP_SCRIPT_VERSION = "1.6.3"
# Number of minutes to check before running script again
$PUBCHECKMIN = 2
$SUBCHECKMIN = 2
$DISCHECKMIN = 2
# Sleep is in seconds not minutes
$RESYNCSLEEP = 120
# Set number of Result logs to keep
$LOGKEEPCNT = 5
# Cross Group Transactions Flag
$global:CROSS_GROUP_TRANSACTION = $false
# Set Exception Log File Names
$reportfilename = $UVHOME + "\RepExcept.errlog"
$scriptrunningname = $UVHOME + "\RepExcept.running"
$scriptruncountfile = $UVHOME + "\RepExcept.cnt"
$scriptworkinfile = $UVHOME + "\RepExcept.inwrk"
$scriptworkoutfile = $UVHOME + "\RepExcept.outwrk"
$scriptworkgrouplist = $UVHOME + "\RepExcept.grplist"
```

```
# Result Variables set in Functions
$global:allsuspended = $true
$global:allsynced = $true
# Get Machine Name
$machinename = [string](hostname)
# Check -v switch
if ($v.IsPresent) {
      Write-Output "UniVerse Home $UVHOME"
      Write-Output "UniVerse Bin $UVBIN"
      Write-Output "UniVerse Version $U2VERSION"
      Write-Output "Machine Name $machinename"
      Write-Output "Script Version $REP_EXCEP_SCRIPT_VERSION"
      exit 0
}
# Declare Send-Email Function
function Send-Email {
      $Body = ""
      $regex = ""
      foreach ($line in Get-Content $reportfilename) {
            if ($line -match $regex) {
                  $Body = $Body + $line + "`r`n"
            }
      }
      "To: " + [Variables]::EmailTo >> $reportfilename
      "Subject: " + $Subject >> $reportfilename
      try {
            $SMTPClient = New-Object
Net.Mail.SmtpClient([Variables]::SMTPServer, [Variables]::SMTPPort)
            $SMTPClient.EnableSsl = [Variables]::SMTPSSL
            $SMTPClient.Credentials = New-Object
System.Net.NetworkCredential([Variables]::SMTPUserName,
[Variables]::SMTPPassword);
            $SMTPClient.Send([Variables]::EmailFrom, [Variables]::EmailTo,
$Subject, $Body)
      }
      catch {
            "Failed to send email" >> $reportfilename
            $_.Exception.GetType().FullName + " " + $_.Exception.Message >>
$reportfilename
      }
      "" >> $reportfilename
}
# Declare Function to remove running file check
function Remove-Run-File {
      try {
            Remove-Item -Path $scriptrunningname
      }
      catch {
            Write-Output "Exception Action Powershell - Unable to remove
$scriptrunningname"
```

```
            $Subject = "U2Replication $machinename - Unable to remove
$scriptrunningname"
            Send-Email
            exit 6
      }
}
# Declare Function to get group names
function Get-Group-Names {
      "Running uv_repadmin report -detail $UNIREP_REMOTESYS to get list of
groups" >> $reportfilename
      uv_repadmin report -detail $UNIREP_REMOTESYS > $scriptworkoutfile
      "" >> $reportfilename
      "Group Names" >> $reportfilename
      $grpcnt = 0
      foreach ($line in Get-Content $scriptworkoutfile) {
            if ($line -match $regex) {
                  $tline = $line.Trim()
                  if ($tline -ne "") {
                        $words = $tline.Split(" ")
                        if (($words[0] -eq "GROUP") -and ($words[2] -eq
"DISTRIBUTION")) {
                              $groupname = $words[1].Substring(0,
$words[1].Length - 1)
                              $groupname >> $reportfilename
                              $grpcnt++
                              if ($grpcnt -eq 1) {
                                    $groupname > $scriptworkgrouplist
                     } else {
                                    $groupname >> $scriptworkgrouplist
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
}
# Declare Function to check that each group is suspended
function Check-All-Suspended {
      "Checking to see if all the groups are Suspended" >> $reportfilename
      $trystart = 1
      foreach ($line in Get-Content $scriptworkgrouplist) {
            if ($line -match $regex) {
                  $groupname = $line
                  for ($trycnt = $trystart; $trycnt -le 10; $trycnt++) {
                        "Running uv_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS" >> $reportfilename
                        uv_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS
                        "Return Code was $LASTEXITCODE" >> $reportfilename
                        if ($LASTEXITCODE -gt $SUB_RESYNCING) {
```

```
                              "Report command failed, exiting powershell" >>
$reportfilename
                              Write-Output "Exception Action Powershell -
uv_repadmin encountered an error and terminated"
                              $Subject = "U2Replication $machinename -
uv_repadmin report encountered an error"
                              Send-Email
                              Remove-Run-File
                              exit 7
                  } else {
                              if (($LASTEXITCODE -eq $REP_SUSPENDED) -or
($LASTEXITCODE -eq $SUB_STOP) -or ($LASTEXITCODE -eq $SUB_EXIT)) {
                                    "Group is suspended" >> $reportfilename
                                    $trycnt = 11
                     } else {
                                    if ($trycnt -eq 10) {
                                          $global:allsuspended = $false
                                          "Could not confirm Suspension of
group, number of retry attempts exceeded" >> $reportfilename
                           } else {
                                          "Could not confirm Suspension of
group, retry attempt $trycnt, sleep 5 seconds and retry" >> $reportfilename
                                          Start-Sleep -Seconds 5
                                          $trystart += 1
                                    }
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
      showud >> $reportfilename
      "" >> $reportfilename
}
# Declare Function to perform sync attempt to recover suspension
function Do-Sync {
      # Check for Transactions as if Transactions are present then we cannot
sync group by group until bug is fixed that stops it from working
      if (-Not($global:CROSS_GROUP_TRANSACTION)) {
            "Checking to see Transactions are being used as indicator not
set" >> $reportfilename
            6 > $scriptworkinfile
            1 >> $scriptworkinfile
            0 >> $scriptworkinfile
            0 >> $scriptworkinfile
            Get-Content $scriptworkinfile | uvreptool > $scriptworkoutfile
            foreach ($line in Get-Content $scriptworkoutfile) {
                  if ($line -match $regex) {
                        $tline = $line.Trim()
                        if ($tline -ne "") {
```

```
                              $words = $tline.Split(" ")
                              if (($words[0] -eq "TCR") -and ($words[1] -eq
"Created:")) {
                                    $numtcr = $words[2]
                                    if (-Not($numtcr -eq "0")) {
                                          $global:CROSS_GROUP_TRANSACTION =
$true
                                          "Transactions detected" >>
$reportfilename
                                    }
                              }
                        }
                  }
            }
      }
      "" >> $reportfilename
      if ($global:CROSS_GROUP_TRANSACTION) {
            "Running uv_repadmin sync -wait -verbose $UNIREP_REMOTESYS" >>
$reportfilename
            uv_repadmin sync -wait -verbose $UNIREP_REMOTESYS >>
$reportfilename
   } else {
            foreach ($line in Get-Content $scriptworkgrouplist) {
                  $groupname = $line
                  "Running uv_repadmin sync -wait -verbose GROUP $groupname
DISTRIB $UNIREP_REMOTESYS" >> $reportfilename
                  uv_repadmin sync -wait -verbose GROUP $groupname DISTRIB
$UNIREP_REMOTESYS >> $reportfilename
            }
      }
      # Check to see if all groups were sync'd okay
      foreach ($line in Get-Content $scriptworkgrouplist) {
            $groupname = $line
            "Running uv_repadmin report GROUP $groupname DISTRIB
$UNIREP_REMOTESYS" >> $reportfilename
            uv_repadmin report GROUP $groupname DISTRIB $UNIREP_REMOTESYS >>
$reportfilename
            "Return Code was $LASTEXITCODE" >> $reportfilename
            if (($LASTEXITCODE -eq $REP_SYNCING) -or ($LASTEXITCODE -eq
$REP_DO_SYNC) -or ($LASTEXITCODE -eq $REP_RUNNING) -or ($LASTEXITCODE -eq
$SUB_RUNNING) -or ($LASTEXITCODE -eq $SUB_SYNCING) -or ($LASTEXITCODE -eq
$SUB_RESYNCING) -or ($LASTEXITCODE -lt $REP_DISABLED)) {
                  $groupsyncing = $true
                  "Group is Syncing or Sync'd" >> $reportfilename
        } else {
                  $global:allsynced = $false
                  "Group is not Syncing or Sync'd" >> $reportfilename
            }
      }
      "" >> $reportfilename
```

```
}
Function Copy-Last-Run-Output {
      Try {
            $lastruncount = Get-Content $scriptruncountfile
      }
      Catch {
            $lastruncount = 0
            $lastruncount > $scriptruncountfile
      }
      $lastruncount = [int]($lastruncount)
      $lastruncount++
      $nextruncount = $lastruncount
      if ($nextruncount -gt $LOGKEEPCNT) {
            $nextruncount = 1
      }
      $nextruncount > $scriptruncountfile
      $copyname = $reportfilename + [string]$nextruncount
      Try {
            Copy-Item -Path $reportfilename -Destination $copyname
      }
      Catch {
            Write-Output "Exception Action Powershell - Unable to copy
$reportfilename to $copyname"
      }
}
# Check to see if Script is Already Running
if (Test-Path -Path $scriptrunningname) {
      Write-Output "Exception Action Powershell - Found $scriptrunningname -
This indicates the exception script is currently running"
      $Subject = "U2Replication $machinename - Found $scriptrunningname -
This indicates the exception script is currently running"
      Send-Email
      exit 4
}
# Check to see if script is running from Replication Exception and get
Replication Environment Variables
if ($d.IsPresent) {
      # If Debug Flag is in place - bypass Replication Environment Check
      $UNIREP_REMOTESYS = [Variables]::DebugRemoteSys
      $UNIREP_REPTYPE = ""
      $UNIREP_GRPNAME = ""
      $UNIREP_ERRCODE = ""
      $UNIREP_ERRSTRING = ""
} else {
      Try {
            $UNIREP_REMOTESYS = Get-Content Env:\UNIREP_REMOTESYS
      }
      Catch {
            Write-Output "Exception Action Powershell - not being run from
Replication Exception"
```

```
            exit 5
      }
      Try {
            $UNIREP_REPTYPE = Get-Content Env:\UNIREP_REPTYPE
      }
      Catch {
            Write-Output "Exception Action Powershell - not being run from
Replication Exception"
            exit 5
      }
      Try {
            $UNIREP_GRPNAME = Get-Content Env:\UNIREP_GRPNAME
      }
      Catch {
            Write-Output "Exception Action Powershell - not being run from
Replication Exception"
            exit 5
      }
      Try {
            $UNIREP_ERRCODE = Get-Content Env:\UNIREP_ERRCODE
      }
      Catch {
            Write-Output "Exception Action Powershell - not being run from
Replication Exception"
            exit 5
      }
      Try {
            $UNIREP_ERRSTRING = Get-Content Env:\UNIREP_ERRSTRING
      }
      Catch {
            Write-Output "Exception Action Powershell - not being run from
Replication Exception"
            exit 5
      }
}
Try {
      $dstring = Get-Date
      # Create Scriptrunning File
      "Run Started at $dstring" > $scriptrunningname
      # If a previous report file exists make a copy of it
      if (Test-Path -Path $reportfilename) {
            Copy-Last-Run-Output
      }
      # Start new report file
      "UniVerse Replication Exception called at $dstring" > $reportfilename
      "Script Version=$REP_EXCEP_SCRIPT_VERSION" >> $reportfilename
      "UniVerse Home=$UVHOME" >> $reportfilename
      "UniVerse Bin=$UVBIN" >> $reportfilename
      "UniVerse Version=$U2VERSION" >> $reportfilename
      "Machine Name=$machinename" >> $reportfilename
```

```
      "" >> $reportfilename
      "Replication Environment Variables" >> $reportfilename
      "UNIREP_REMOTESYS=$UNIREP_REMOTESYS" >> $reportfilename
      "UNIREP_REPTYPE=$UNIREP_REPTYPE" >> $reportfilename
      "UNIREP_GRPNAME=$UNIREP_GRPNAME" >> $reportfilename
      "UNIREP_ERRCODE=$UNIREP_ERRCODE" >> $reportfilename
      "UNIREP_ERRSTRING=$UNIREP_ERRSTRING" >> $reportfilename
      "" >> $reportfilename
      "Replication Status Error Codes" >> $reportfilename
      "PUB_STOP=$PUB_STOP" >> $reportfilename
      "PUB_EXIT=$PUB_EXIT" >> $reportfilename
      "PUB_SHUTDOWN=$PUB_SHUTDOWN" >> $reportfilename
      "PUB_RUNNING=$PUB_RUNNING" >> $reportfilename
      "PUB_DO_RECONFIG=$PUB_DO_RECONFIG" >> $reportfilename
      "PUB_DO_BACKSYNC=$PUB_DO_BACKSYNC" >> $reportfilename
      "PUB_DO_FAILOVER=$PUB_DO_FAILOVER" >> $reportfilename
      "REP_RUNNING=$REP_RUNNING" >> $reportfilename
      "REP_SUSPENDED=$REP_SUSPENDED" >> $reportfilename
      "REP_SYNCING=$REP_SYNCING" >> $reportfilename
      "REP_DO_SUSPEND=$REP_DO_SUSPEND" >> $reportfilename
      "REP_DO_SYNC=$REP_DO_SYNC" >> $reportfilename
      "REP_EXIT=$REP_EXIT" >> $reportfilename
      "REP_DISABLED=$REP_DISABLED" >> $reportfilename
      "SUB_STOP=$SUB_STOP" >> $reportfilename
      "SUB_EXIT=$SUB_EXIT" >> $reportfilename
      "SUB_SHUTDOWN=$SUB_SHUTDOWN" >> $reportfilename
      "SUB_RUNNING=$SUB_RUNNING" >> $reportfilename
      "SUB_DO_RECONFIG=$SUB_DO_RECONFIG" >> $reportfilename
      "SUB_DO_SUSPEND=$SUB_DO_SUSPEND" >> $reportfilename
      "SUB_SYNCING=$SUB_SYNCING" >> $reportfilename
      "SUB_DO_FAILOVER=$SUB_DO_FAILOVER" >> $reportfilename
      "SUB_RESYNCING=$SUB_RESYNCING" >> $reportfilename
      "" >> $reportfilename
      cd $UVBIN
      showud >> $reportfilename
      "" >> $reportfilename
      $publisher = $false
      $subscriber = $false
      $disabled = $false
      $unexpectedcode = $false
      "Running uv_repadmin report -detail $UNIREP_REMOTESYS" >>
$reportfilename
      uv_repadmin report -detail $UNIREP_REMOTESYS >> $reportfilename
      "" >> $reportfilename
      "Is this system a publisher or subscriber or is replication disabled ?
Exit Code was $LASTEXITCODE" >> $reportfilename
      if ($LASTEXITCODE -lt $REP_DISABLED) {
            $publisher = $true
            "System is a publisher - No Recovery will take place" >>
$reportfilename
```

```
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Publisher)"
      }
      if ($LASTEXITCODE -gt $REP_DISABLED) {
            $subscriber = $true
            "System is a subscriber" >> $reportfilename
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Subscriber)"
      }
      if ($LASTEXITCODE -eq $REP_DISABLED) {
            $disabled = $true
            "Replication is disabled" >> $reportfilename
            $Subject = "U2Replication $machinename - REPLICATION DISABLED"
      }
      if ($LASTEXITCODE -gt $SUB_RESYNCING) {
            $unexpectedcode = $true
            $subscriber = $false
            "Return code is outside expected Range, so the assumption the
System is a subscriber may be incorrect - No recovery will be performed" >>
$reportfilename
            $Subject = "U2Replication $machinename - REPLICATION SUSPENDED
(Subscriber with Unexpected Code)"
      }
      "" >> $reportfilename
      "Sending Email Notification of Suspension" >> $reportfilename
      "" >> $reportfilename
      Send-Email
      # Only Run Recovery Steps if System is a subscriber
      if ($subscriber) {
            Get-Group-Names
            Check-All-Suspended
            if ($global:allsuspended) {
                  Write-Output "Exception Action Powershell - All groups
suspended - Powershell Sleep for $RESYNCSLEEP seconds"
                  $Subject = "U2Replication $machinename - Confirmed
Suspension of the all Groups - Resync in 2 Minutes (Subscriber)"
                  "Sleeping for $RESYNCSLEEP seconds before sync attempt" >>
$reportfilename
                  Send-Email
                  Start-Sleep -Seconds $RESYNCSLEEP
                  "Woken up from Sleep" >> $reportfilename
                  "" >> $reportfilename
                  Do-Sync
                  If ($global:allsynced) {
                        Write-Output "Exception Action Powershell - Confirmed
Sync of the all Groups"
                        $Subject = "U2Replication $machinename - Confirmed
Sync of the all Groups (Subscriber)"
        } else {
```

```
                        Write-Output "Exception Action Powershell - Unable to
Confirm Sync of the all Groups"
                        $Subject = "U2Replication $machinename - Unable to
Confirm Sync of the all Groups (Subscriber)"
                  }
                  Send-Email
   } else {
                  Write-Output "Exception Action Powershell - Unable to
confirm all groups suspended"
                  $Subject = "U2Replication $machinename - Unable to Confirm
Suspension of the all Groups - No Resync (Subscriber)"
                  Send-Email
            }
      }
}
Catch {
      Remove-Run-File
      Write-Output $_.Exception|format-list -force
      Write-Output "Exception Action Powershell - An Error Occurred"
      exit 8
}
Remove-Run-File
Write-Output "Exception Action Powershell - Reached Normal End of Exception
Action Powershell"
exit 0
```

Chapter 2: Introduction to the Monitor Phantoms

The Monitor Phantoms were also developed as a proactive monitoring tool for U2 Replication and are again designed to send emails as an advance warning of events that need an Administrators attention before they develop into a larger problem. Although the monitor phantoms are designed to and will only run on a subscriber system, there are components such as the replication configuration checker and performance monitoring than can also run on a publisher. Therefore, the suite of programs can be installed on both the publisher and subscriber systems.

The Monitor Phantoms are deployed into an account of your choice. Our recommendation would be to be a create a new account specifically for this purpose and ensure the account itself is not replicated. All the source code for the monitor phantoms will be left on the subscribing system for your review and addition. The source code deployment and setup is done via a VB.NET program that we provide. The source code once deployed will be stored in a file ROCKET.BP.

We recommend the program is installed into C:\deploy on a client PC and then the application can be installed from the RepPhantoms subdirectory using the Microsoft Click-Once deployment.

Our current recommendation is to run with RW_IGNORE_ERROR set to 1. With RW_IGNORE_ERROR set to 1 then if a replication writer process encounters an error the error is logged to the replication writer error log file (rw.errlog on UniData and uvrw.errlog on UniVerse) without suspending replication. If RW_IGNORE_ERROR is set to 0 then replication is suspended each time a replication writer process encounters an error.

Following our recommendation of RW_IGNORE_ERROR to be set to 1 then unless the error log file is regularly checked, errors could go unnoticed. The purpose of one of the phantoms is to scan the error log at defined intervals and then send email notifications to a defined list of recipients.

The purpose of the second phantom is to monitor the LSN (Logical Sequence Number) of U2 Replication updates. Each record update in each replication group is given a sequentially issued LSN reference. As the update moves through the replication system it's marked as Pub Done, Sub Got, Sub Available and Sub Done. The Replication Monitor Tool in XAdmin shows these counts and unless the monitor is regularly viewed any potential problems could go unnoticed for a period time.

The second phantom takes a snap shot of the LSN counts at defined intervals. It then checks those counts and the overall group status to the previous counts.

- If the overall group status is anything other than SUB_SYNCING or SUB_RUNNING the group will be reported to be in error.
- If all the LSN Counts are equal then the group is not considered to be in error.
- If the LSN Counts indicate a stall in processing in the subscriber receiving or applying updates then the group will be reported to be in error.

The second phantom could be updated to also report on slowdowns of LSN updates or to report if a subscriber is falling further behind a publisher on each iteration. We do provide the source code for the server side programs for your review and addition.

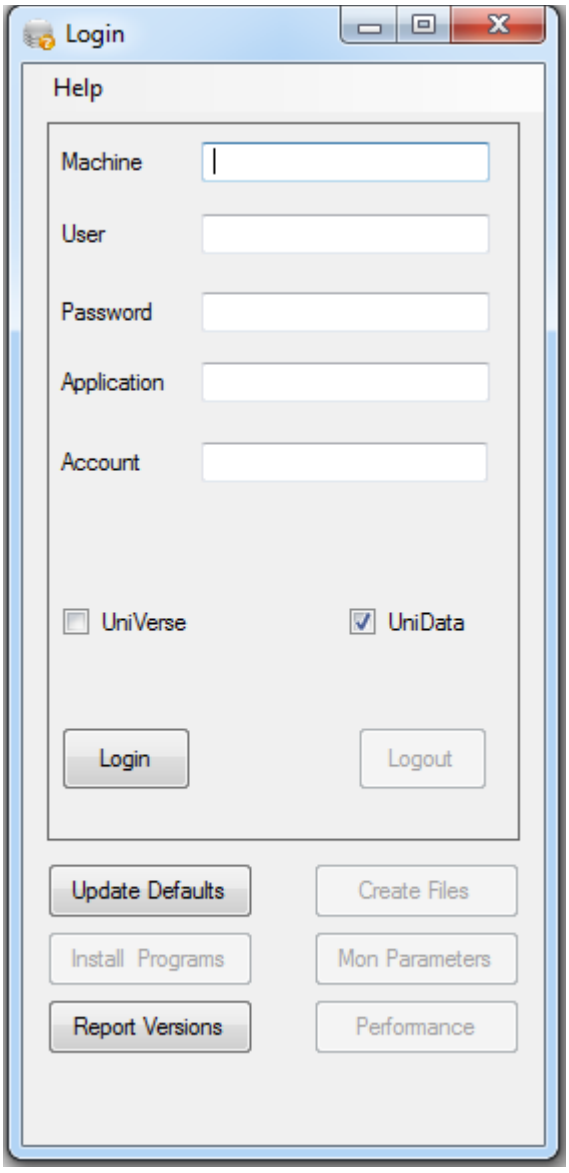
The phantoms were developed to use mailx/sendmail in Unix as the email client. On windows we're using a smtp gmail account and internally we've uses a second VB.NET program to send the emails. We will not be assisting in helping you to setup mailx/sendmail or gmail or develop VB.NET programs to send email messages.

Starting at version 1.4.1 the performance monitoring / gathering screen was added. This screen enables you to turn on and off the performance monitoring capabilities of U2 Replication. The gathering function gathers the performance statistics and places them into server 'csv' files that can be imported into Excel for further analysis.

Using the VB.NET Deployment Program

When you first use the program the following screen will be displayed.

Login Form



You are then required to complete the details that are required to connect to the subscribing system account that you wish to use to install the monitor programs. We currently recommend that this account is not part of U2 Replication and this will avoid extra potential reconfiguration of U2 Replication that may be needed to get the programs to install.

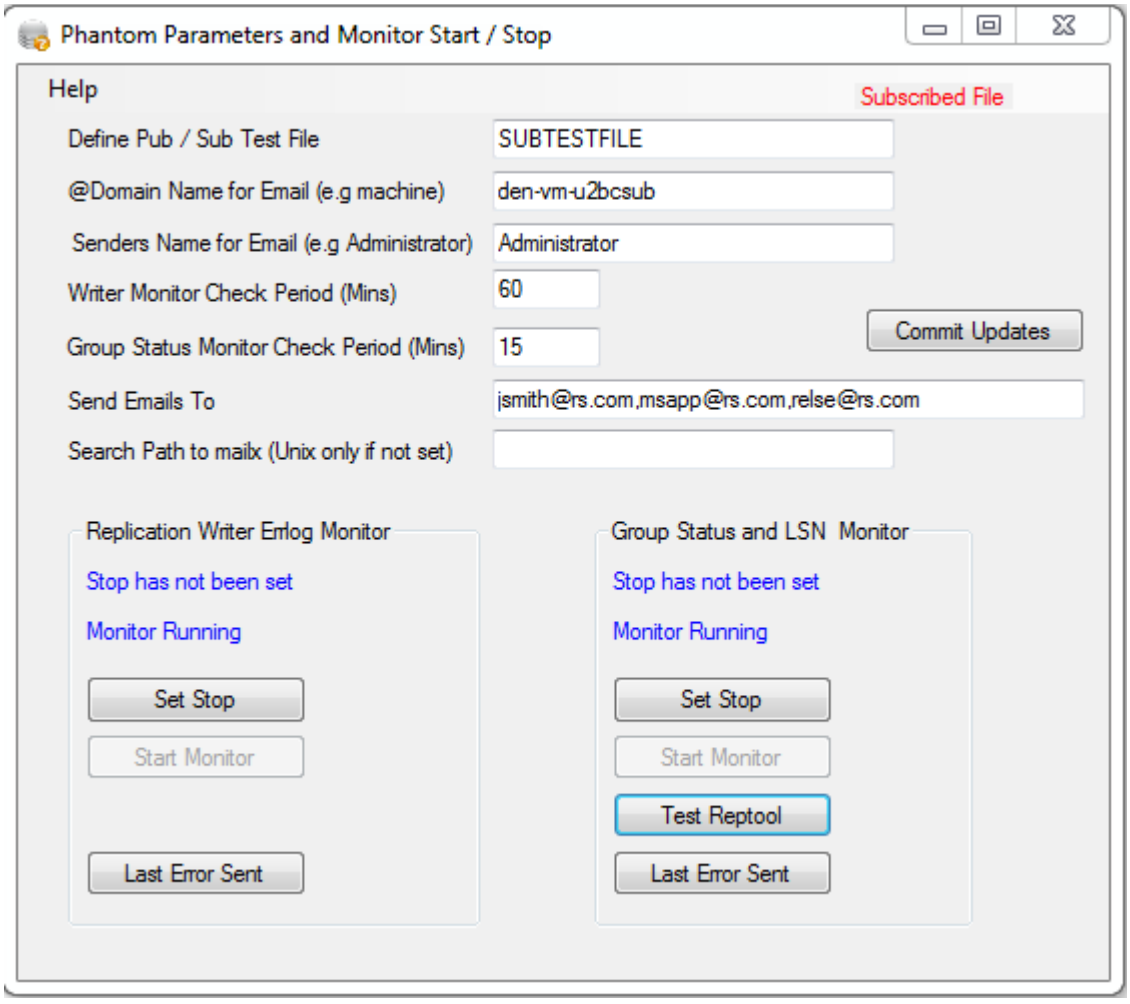
- Machine - Enter the name or the ip address of the machine you wish to connect to install or maintain the U2 Replication Monitor Phantoms.
- User - The name of the user that will be used to connect to the machine.
- Password - The password for the user that will used to connect to the machine.
- Application - (UniData Only and Optional) UniData provides an extra level of security in that an optional Application Name can be provided to the UO Connection. If this is being used then the name of the UniObjects Application Control can be provided here.
- Account - The name of the account you wish to use to maintain and run the monitor phantoms.
    - On UniVerse this can be the name as specified in UV.ACCOUNT file or the path name to the account.
    - On UniData this has to be the path name to the account.
- UniVerse Check Box - Check this box if you wish to connect to UniVerse on the machine.
- UniData Check Box - Check this box if you wish to connect to UniData on the machine.
- Login Button - This button will become available when you are allowed to Login to the machine with the connection credentials as defined in the form. Once you've successfully Logged In the button will be disabled.
- Logout Button - Initially disabled this button will become active when you've successfully logged in. When enabled it allows you to Logout / Disconnect from the machine.
- Report Versions Button – This Button will the versions of Client Software, Master Software / Scripts and the versions of the Software / Scripts installed on the server.
- Mon Parameters Button - This button when enabled allows the Parameters form where the parameters for the Monitors can be maintained. The Monitors can also be stopped and started from the Parameters Screen.
- Update Defaults Button - This button allows you to store your login credentials (minus the password) to be used as the default values when you enter the login screen again.
- Create Files Button - This button will become available if the application detects that the ROCKET.BP / REP.PARAMS file do not exist in the account on machine you logged into and when pressed will create those files.

- ***You will need to use the Create Files Button the first time after connecting to a new machine or new account.***

- Install / Upgrade / Reinstall Programs Button - This button when enabled will change its title from Install to Upgrade to Reinstall.
    - When the button is labelled 'Install' the application has detected the programs needed to run the monitor phantoms have yet to be installed on the system and will install them when pressed.
        - ***You will need to use the 'Install' button the first time you've connected to a new account / new machine once you've used the 'Create Files' Button***
    - When the Label is Upgrade the application has detected newer versions of the programs are available and pressing Upgrade will upgrade the programs (Note if the monitor processes are running on the machine they will need to be stopped and restarted to pick up the newer versions of the programs).
    - When the label is 'Reinstall' it allows you to reinstall the programs if required to do so.
    - As part of this process a copy of the example exception scripts for UNIX detailed earlier in this document are also placed in the file ROCKET.BP on the server.
- Performance Button – This button when enabled will allow you to enter the performance monitoring and capture screen
    - Version 1.4.1 and above of the Host Programs need to be installed for this button to be enabled.

Monitor Parameters Form

Once you've successfully connected to the machine, created the files and installed the programs on the server you'll be allowed to enter the Monitor Parameters Form where a screen similar to the one below will be displayed. The screen below is one from one of our internal machines.



- Define Pub / Sub Test File - This is the name of VOC pointer in the target account that should point to a replicated file. The replication status of that file is then examined to determine the publisher / subscriber status of the machine. This file has to be a 'Subscribed File' for all of the functionality to work correctly.
- @Domain Name and Senders Name - These values will be combined to produce an email address of 'Senders Name @ Domain Name'. This email address will be used by monitoring programs as the 'Senders' email address of any email notification sent out.

- Write Monitor Check Period - This defines how often that the 'Replication Writer Error Log Monitor' will scan the Error Log File. It is defined in minutes. A value of 30 would result in the Error Log being scanned every 30 minutes.

- Group Status Monitor Check Period - This defines how often that the 'Group Status and LSN Monitor' will scan each replication. It is defined in minutes. A value of 15 would result in each group being scanned every 15 minutes.

- Send Emails To - This is the list of email recipients that will receive any notification from the monitoring phantoms. On a Unix system this will be a comma delimited list as the default mail client of mailx uses that.

    - If you change to a different mail client then the values entered here will need to confirm with that change and you will also need to modify the monitoring programs in ROCKET.BP accordingly.

    - For Windows systems please you'll need to develop your own extra emailing program.

- Search Path to Mailx - (UNIX Only) If the path to the mailx executable is not defined in the default path of the user running the monitor phantoms then the path can be fully defined here.

- Commit Updates Button - When pressed this Button will write the parameters back to the REP.PARAMS file on the machine.

- Replication Writer Errlog Monitor - This will display two messages in relation to the status of the Monitor Phantom.

    - The first message will read 'Stop Not Set' or 'Stop Set'. A message of 'Stop Not Set' indicates that no 'STOP' flag is present in the REP.PARAMS file for the Monitor. A Message of 'Stop Set' indicates that the 'STOP' flag has been set in the REP.PARAMS file and the monitor process will check that flag during it's next checking period and then close down.

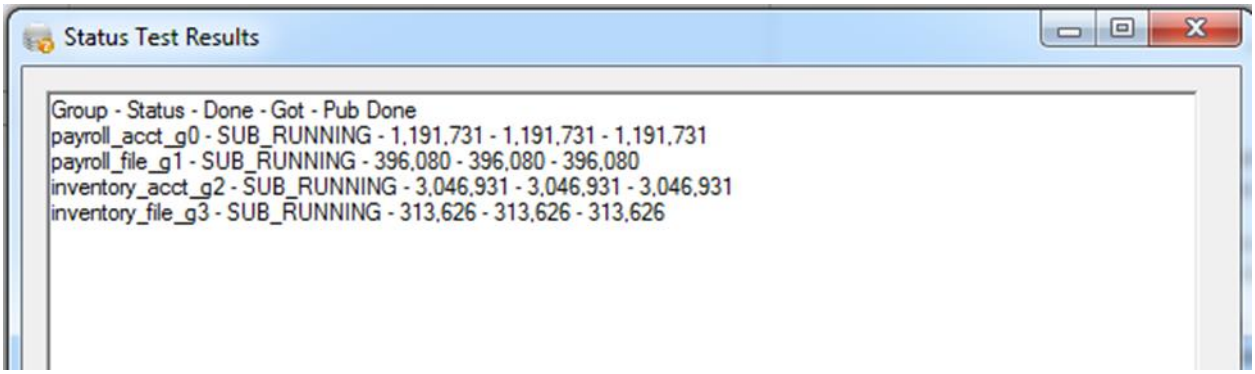
    - The second message will read 'Monitor Not Running' or 'Monitor Running'. A message of 'Monitor Not Running' indicates that the monitor process could not be found in the current list of processes using the PORT.STATUS verb. Similarly, a message of 'Monitor Running' indicates the monitor process was found in the current list of processes.

    - The status of each message controls the availability of the other buttons in the container.

        - Set Stop Button - This button when enabled and pressed will set the 'STOP' flag in the REP.PARAMS file for the process to check during the next checking period.

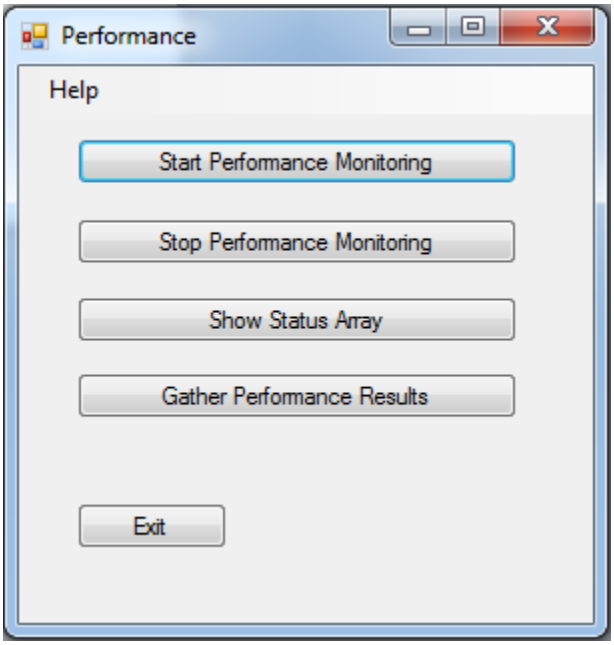
        - Start Monitor Button - This button when enabled and pressed will send a request to the server to start the monitoring process as a phantom process on the machine.

- Last Error Sent – This will display the text used for the last email message that was generated for sending
- Group Status and LSN Monitor - For an explanation of the first two messages and two buttons see the above on the 'Replication Writer Errlog Monitor'.
- Test Reptool - This button when enabled and pressed allows a test of the programs on the server that interface with the reptool / uvreptool command. If successful then it will return the Status together with the Pub Done,Sub Done and Sub Got LSN Counters to a message box.
    - If these are not returned as above then an error has occurred in that interface and needs to be investigated.

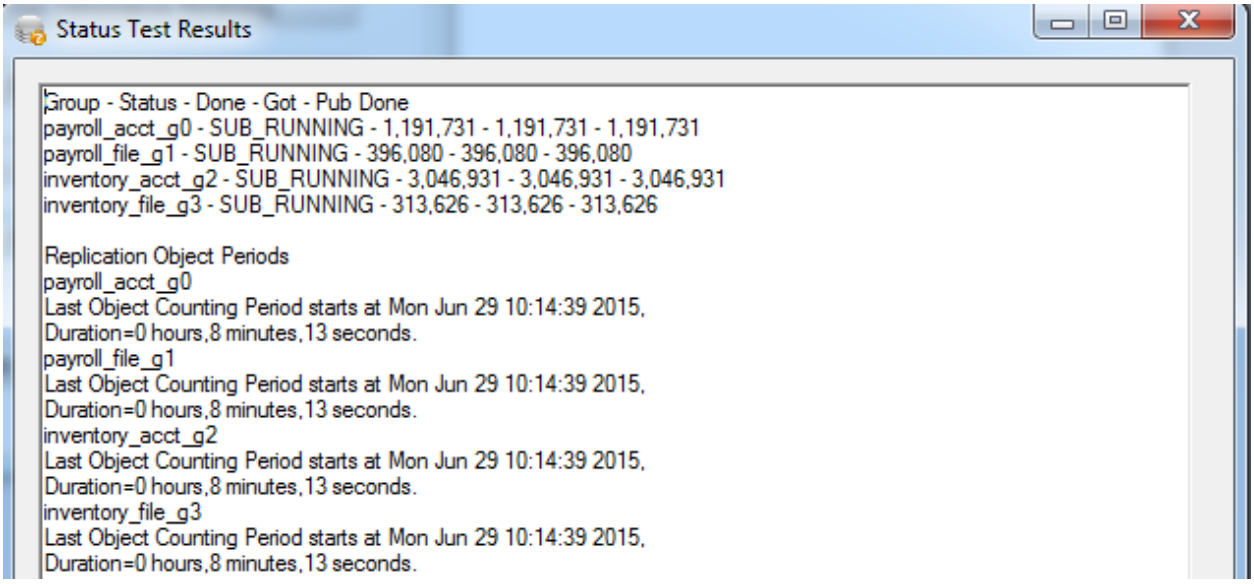


Performance Form

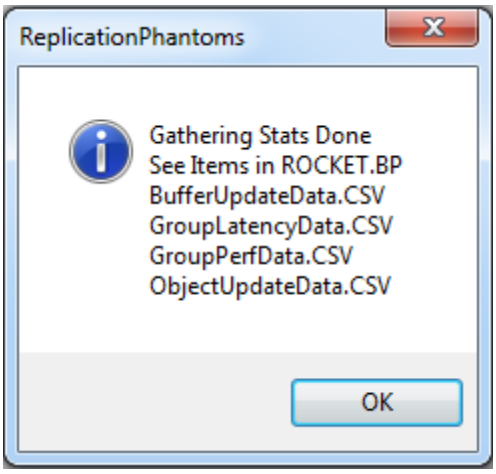
Once you've successfully connected to the machine, created the files and installed version 1.4.1 or above of the programs on the server you'll be allowed to enter the Performance Form where a screen like the one below will be displayed. In terms of gathering the performance data this can be done on the publisher or subscriber and in our experience is more useful if taken from a publishing system.

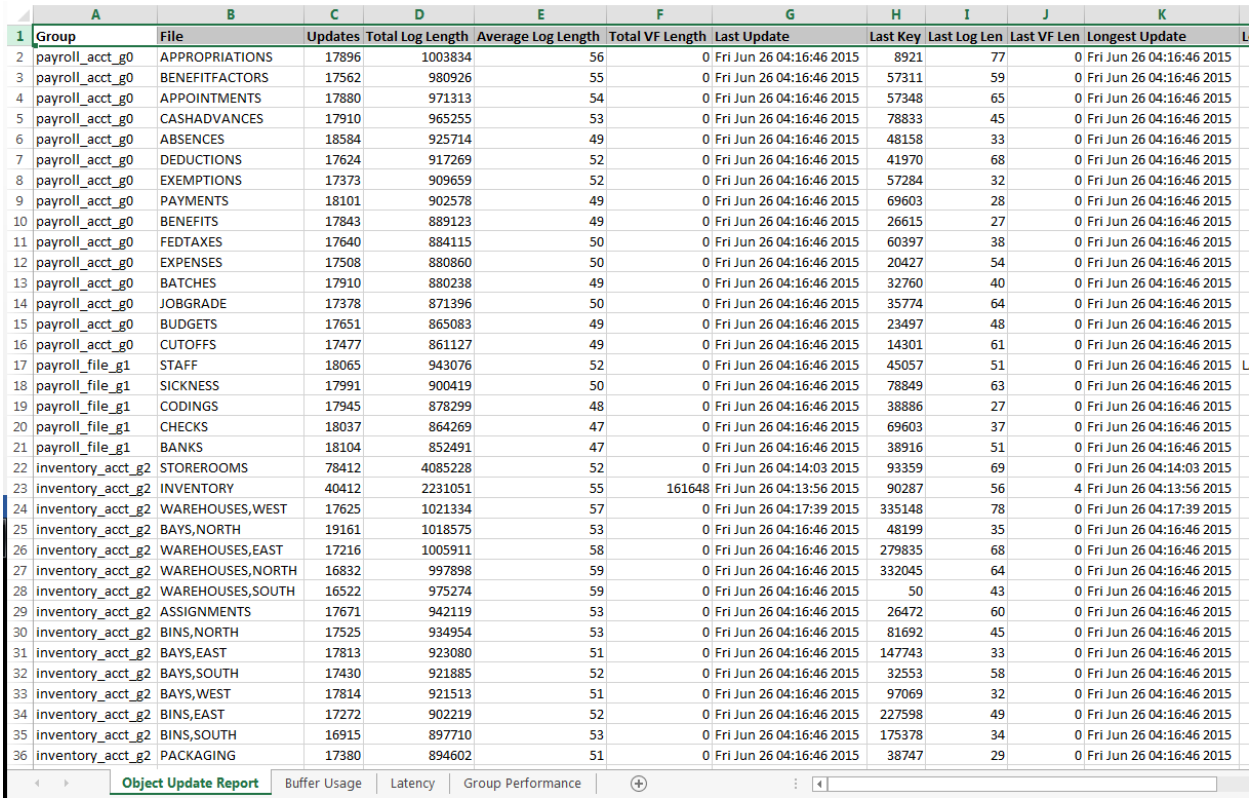


- Start Performance Monitoring – This will start the performance monitoring part of replication on the server and start to gather performance data.
- Stop Performance Monitoring – This will stop the performance monitoring part of replication on the server and stop the gathering of performance data.
- Show Status Array – This is a test function and will display the last available overview of the performance data. If no monitoring has been done before or a monitoring sequence is currently active not all of the information will be available (See screenshot on the next page)



- Gather Performance Results – This will combine all the performance data into four CSV files located in ROCKET.BP on the server. These files can be subsequently uploaded onto a client PC and used in tools such as excel for further analysis of the information
    - BufferUpdateData.CSV contains the Replication Buffer Usage during the test period
    - GroupLatencyData.CSV contains the latency information during the test period
    - GroupPerfData.CSV contains the group performance data during the test period
    - ObjectUpdateData.CSV contains the details about each file during the test period.





	A	B	C	D	E	F	G	H	I	J	K
1	Group	File	Updates	Total Log Length	Average Log Length	Total VF Length	Last Update	Last Key	Last Log Len	Last VF Len	Longest Update
2	payroll_acct_g0	APPROPRIATIONS	17896	1003834	56	0	Fri Jun 26 04:16:46 2015	8921	77	0	Fri Jun 26 04:16:46 2015
3	payroll_acct_g0	BENEFITFACTORS	17562	980926	55	0	Fri Jun 26 04:16:46 2015	57311	59	0	Fri Jun 26 04:16:46 2015
4	payroll_acct_g0	APPOINTMENTS	17880	971313	54	0	Fri Jun 26 04:16:46 2015	57348	65	0	Fri Jun 26 04:16:46 2015
5	payroll_acct_g0	CASHADVANCES	17910	965255	53	0	Fri Jun 26 04:16:46 2015	78833	45	0	Fri Jun 26 04:16:46 2015
6	payroll_acct_g0	ABSENCES	18584	925714	49	0	Fri Jun 26 04:16:46 2015	48158	33	0	Fri Jun 26 04:16:46 2015
7	payroll_acct_g0	DEDUCTIONS	17624	917269	52	0	Fri Jun 26 04:16:46 2015	41970	68	0	Fri Jun 26 04:16:46 2015
8	payroll_acct_g0	EXEMPTIONS	17373	909659	52	0	Fri Jun 26 04:16:46 2015	57284	32	0	Fri Jun 26 04:16:46 2015
9	payroll_acct_g0	PAYMENTS	18101	902578	49	0	Fri Jun 26 04:16:46 2015	69603	28	0	Fri Jun 26 04:16:46 2015
10	payroll_acct_g0	BENEFITS	17843	889123	49	0	Fri Jun 26 04:16:46 2015	26615	27	0	Fri Jun 26 04:16:46 2015
11	payroll_acct_g0	FEDTAXES	17640	884115	50	0	Fri Jun 26 04:16:46 2015	60397	38	0	Fri Jun 26 04:16:46 2015
12	payroll_acct_g0	EXPENSES	17508	880860	50	0	Fri Jun 26 04:16:46 2015	20427	54	0	Fri Jun 26 04:16:46 2015
13	payroll_acct_g0	BATCHES	17910	880238	49	0	Fri Jun 26 04:16:46 2015	32760	40	0	Fri Jun 26 04:16:46 2015
14	payroll_acct_g0	JOBGRADE	17378	871396	50	0	Fri Jun 26 04:16:46 2015	35774	64	0	Fri Jun 26 04:16:46 2015
15	payroll_acct_g0	BUDGETS	17651	865083	49	0	Fri Jun 26 04:16:46 2015	23497	48	0	Fri Jun 26 04:16:46 2015
16	payroll_acct_g0	CUTOFFS	17477	861127	49	0	Fri Jun 26 04:16:46 2015	14301	61	0	Fri Jun 26 04:16:46 2015
17	payroll_file_g1	STAFF	18065	943076	52	0	Fri Jun 26 04:16:46 2015	45057	51	0	Fri Jun 26 04:16:46 2015
18	payroll_file_g1	SICKNESS	17991	900419	50	0	Fri Jun 26 04:16:46 2015	78849	63	0	Fri Jun 26 04:16:46 2015
19	payroll_file_g1	CODINGS	17945	878299	48	0	Fri Jun 26 04:16:46 2015	38886	27	0	Fri Jun 26 04:16:46 2015
20	payroll_file_g1	CHECKS	18037	864269	47	0	Fri Jun 26 04:16:46 2015	69603	37	0	Fri Jun 26 04:16:46 2015
21	payroll_file_g1	BANKS	18104	852491	47	0	Fri Jun 26 04:16:46 2015	38916	51	0	Fri Jun 26 04:16:46 2015
22	inventory_acct_g2	STOREROOMS	78412	4085228	52	0	Fri Jun 26 04:14:03 2015	93359	69	0	Fri Jun 26 04:14:03 2015
23	inventory_acct_g2	INVENTORY	40412	2231051	55	161648	Fri Jun 26 04:13:56 2015	90287	56	4	Fri Jun 26 04:13:56 2015
24	inventory_acct_g2	WAREHOUSES,WEST	17625	1021334	57	0	Fri Jun 26 04:17:39 2015	335148	78	0	Fri Jun 26 04:17:39 2015
25	inventory_acct_g2	BAYS,NORTH	19161	1018575	53	0	Fri Jun 26 04:16:46 2015	48199	35	0	Fri Jun 26 04:16:46 2015
26	inventory_acct_g2	WAREHOUSES,EAST	17216	1005911	58	0	Fri Jun 26 04:16:46 2015	279835	68	0	Fri Jun 26 04:16:46 2015
27	inventory_acct_g2	WAREHOUSES,NORTH	16832	997898	59	0	Fri Jun 26 04:16:46 2015	332045	64	0	Fri Jun 26 04:16:46 2015
28	inventory_acct_g2	WAREHOUSES,SOUTH	16522	975274	59	0	Fri Jun 26 04:16:46 2015	50	43	0	Fri Jun 26 04:16:46 2015
29	inventory_acct_g2	ASSIGNMENTS	17671	942119	53	0	Fri Jun 26 04:16:46 2015	26472	60	0	Fri Jun 26 04:16:46 2015
30	inventory_acct_g2	BINS,NORTH	17525	934954	53	0	Fri Jun 26 04:16:46 2015	81692	45	0	Fri Jun 26 04:16:46 2015
31	inventory_acct_g2	BAYS,EAST	17813	923080	51	0	Fri Jun 26 04:16:46 2015	147743	33	0	Fri Jun 26 04:16:46 2015
32	inventory_acct_g2	BAYS,SOUTH	17430	921885	52	0	Fri Jun 26 04:16:46 2015	32553	58	0	Fri Jun 26 04:16:46 2015
33	inventory_acct_g2	BAYS,WEST	17814	921513	51	0	Fri Jun 26 04:16:46 2015	97069	32	0	Fri Jun 26 04:16:46 2015
34	inventory_acct_g2	BINS,EAST	17272	902219	52	0	Fri Jun 26 04:16:46 2015	227598	49	0	Fri Jun 26 04:16:46 2015
35	inventory_acct_g2	BINS,SOUTH	16915	897710	53	0	Fri Jun 26 04:16:46 2015	175378	34	0	Fri Jun 26 04:16:46 2015
36	inventory_acct_g2	PACKAGING	17380	894602	51	0	Fri Jun 26 04:16:46 2015	38747	29	0	Fri Jun 26 04:16:46 2015

Object Update Report | Buffer Usage | Latency | Group Performance

Disaster Recovery Notice

Designing, building, implementing and maintaining your Disaster Recovery solution is very complex, including server, firmware, storage, and networking. Configuring and implementing replication to meet your Disaster Recovery Service Level Agreements (SLAs) and requirements, while considering the performance and current SLAs in your production environment requires planning and experience. U2 Replication is not a product where you should "Teach Yourself". Disaster Recovery is not a place to "Learn on the Job". Inadvertent omissions and mistakes during implementation may have severe consequences for your data and your organization.

Please include Rocket U2 Education and Rocket Solutioning Services in your plan for success.

There are five products Rocket Software does not recommend that you implement by yourself without Rocket U2 Education for your staff or assistance Rocket Solutioning Services:

1. Automatic Data Encryption (ADE)
2. Recoverable File System (RFS)
3. U2 Audit Logging
4. U2 Replication
5. Web/DE

Rocket U2 Education supplies standard courses to learn U2 Replication. These should be included in your overall plan for success.

We "Highly" recommend you rely upon the experience gained by many installations that Rocket Solutioning Services has performed.

Rocket Solutioning Services offers standard packages of services to assist you in implementing U2 Replication which includes methodologies, project plans, implementation assistance, knowledge transfer to your staff, post "Go Live" assistance, and the "As Built" Document defining how your system was implemented which includes recovery procedures.

The "As Built" Document is distributed to your staff, your Application Provider (if applicable) and the Rocket U2 Support organization. These multiple layers of support all will assist you in getting your system back up when failure occurs. They all need to be informed as to what was implemented.

Should you still wish to attempt this yourself (not recommended), Rocket Solutioning Services offers a "Validation Package" where we review what you have implemented and make recommendations for an effective implementation.

Restating:

Please include Rocket U2 Education and Rocket Solutioning Services in your plan for success.