

Sybase Repserver Notes

Handy tips for the busy DBA

Last updated: 25/04/2017

DBXperts Ltd

Garrett Devine

www.dbxperts.co.uk



Secure Remote Database Administration

Table of Contents

Document Revision 1.8	4
Introduction & Disclaimer	4
Repserver Components.....	4
More Detailed Look at the Components	4
Examine replication environment.....	4
Repserver BASICS	5
General Install	5
Table Defs Install.....	5
Warm Standby Install	6
Warm Standby Switch over	7
Database (MSA) repdef	7
Manually set up connections	8
setup primary db's for rep.....	8
Function Repdefs (stored procedure replication)	9
Replication Tuning Notes	10
Golden rules	10
Find Bottlenecks	10
Sizing caches sizes.....	11
SQT_MAX_CACHE_SIZE	11
DSI_SQT_MAX_CACHE_SIZE	11
Tuning	11
Tuning Primary DB	11
Tuning Rep Server	11
Tuning RSSD.....	11
Tuning Replicate DB	12
Tuning DSI	12
Configure for SMP systems	12
Minimal Column Replication	12
Monitor Counters.....	13
Not requiring Setup.....	13
Requiring Setup	13
Disaster Recovery Notes.....	13
Recover from reloading Primary Database	13
Skipping transactions	14
Stop Replication	14
Replaying Transaction Logs	14
Rebuild a Stable Device - with tran log	14
Rebuild a Stable Device - without tran log	15
Restore the RSSD from backup.....	15
General Troubleshooting	16
Stable Queue Full	16
Ignoring duplicate keys – when we have a lot, use error class!	16
Reverse Engineering an Error Class	17
HowTo determine the error class configured for a connection.....	17
Row count mismatch – use a replication server error class. V15.2+.....	17
Displays all Replication Server configuration parameters.....	18
Determine Latency	18
Dropping Subscriptions Fast	18
Detecting loss	19
Repserver Trace Flags	19
Configure the rep agent to trace LTL--write output to a trace file (not to ASE log)	19

Turn on Rep Agent tracing and DSI/function string tracing	20
Turn off Rep Agent tracing and DSI/function string tracing	20
Handy Tips	20
Renaming a replicate database	20
rs_ticket Feature	21
Setup	21
Exclude rs_ticket_history from replication	21
Send your first 'ticket'	21
Checking the results of your ticket	21
How to implement	22
Get data from the RDB to trend the latency	22
Appendix A – Shell scripts	24
rs_checkreplag.ksh	24
sp_queueinfo	25
rs_ticket.sh	25
Appendix B – troubleshooting	26
Uninstall repserver program.....	26
Logical Connection will not Drop.....	26

Document Revision 1.8

Introduction & Disclaimer

The notes contained in this document are intended as a fast find guide to using Sybase Replication server and have been built up over my time using repserver in the real world. It is not intended to be a complete exploration of all of replication server's abilities, nor do I claim that all the notes are without error. If you find errors or would like to submit your own top tip for the next edition of this guide, then please email us at info@ddsafe.co.uk

Repserver Components

=====

SQM (Stable Queue Manager) to manage inserts/deletes and prevent duplicates. One per Queue

LTM, Log Transfer Manager. Reads the transaction log.

Inbound queue. Holds transactions from LTM. 'admin who, sqm' shows these, e.g. 456:1. the ':1' means inbound

Outbound queue. Holds trans. to be replicated 'admin who, sqm' shows these, e.g. 457:0. the ':0' means outbound. Has 2 types of queue. Data Server Interface (DSI) and Replication Server Interface (RSI), used across routes.

Distributor (DIST). Matches repdefs with subscriptions, so messages applied correctly to replicate. One DIST thread per inbound queue.

SQT (Stable Queue Transaction Manager) ensures queues are accessed in transactional manner. SQT has 4 queues:-

- * Open queue that holds transactions until commit or rollback is read from LTM
- * Closed queue holds completed transactions.
- * Read queue holds data that has been read from the Closed queue and a receipt of the transaction received. Tran is then removed from queue.
- * Truncation queue holds 'begin tran' record. Queue is used to determine which transactions can be deleted.

More Detailed Look at the Components

Admin who, sqm.	First Seg.Block - Last Seg.Block = data in queue (Mb)
Admin who, sqt.	Next Read is the next segment, block & row to be read from queue. First Trans gives queue status. st=status of 1 st command, cmds=no of commands in transaction. qid=seg:block:row where tran starts. Full - if non-zero, sqt_max_cache_size too small.
Sqt_max_cache_size	cache available to SQT. Need 1M per queue. (Ensure value of sqt_max_xcache_size * num. of queues is less than memory_limit)
DIST	matched repdefs with subs. 3 components. SRE: matched repdefs with Subs. TD: packages transactions. MD: delivers messages if routes involved. 'admin who, dist' gives totals of commands processed & ignored.
DSI	reads committed commands (in SQT closed queue) and applies them to replicate DB. Prevents duplicates. Groups transactions to replicate. Grouping defined by dsi_zact_group_size & dsi_cmd_batch_size.
RSI	Routes between repservers across WANs

Examine replication environment

```
sp_setreptable --in pdb
admin rssd_name -- in RS
admin who      -- in RS
admin logical_status -- in RS
rs_helpprep   -- in RSSD
rs_helppdbrep -- in RSSD
```

```

rs_helpdb      -- in RSSD
rs_helproute   -- in RSSD
rs_helpsub     -- in RSSD, details table subscriptions
rs_helpdbsub   -- in RSSD
rs_helppubsub  -- in RSSD, if using publications
rs_helpdbpub   -- in RSSD, details publication subscriptions, articles and subscribers
rs_helpuser    -- in RSSD

```

To look at current connection settings, use 'admin config'.

```

admin config [, [{"connection" | logical_connection}, data_server, database] |
             ["route", repserver]] [, configuration_name]

```

Example:-

```

admin config, "connection", <servername>, <dbname>, dsi_quoted_identifier

```

```

#-----#

```

Repserver BASICS

```

#-----#

```

If you use rs_init to configure replication and it fails, you can sometimes get more information out of the rs_init log files. These are located at \$SYBASE/\$SYBASE_REP/init/logs

General Install

Use rs_init to install repserver & set up the RSSD. Create stable queue files first (using 'touch'). Once this is complete, you need to add connections to the primary and replicate dataservers and databases. See the sections below on how to do this. To use the GUI (rs_init), create a rep maint user in the DB using sp_adduser. Remove this later and add as alias to dbo using sp_addalias.

Table Defs Install

In PDB

```

=====
sp_setreptable prim_tab1, true

```

In RS

```

=====
1> create replication definition prim_tab1_repdef with primary at SRV01_ASE.pdb1
2> with all tables named prim_tab1 (a int, b char(10)) primary key (a)
3> go

1> define subscription prim_tab1_sub for prim_tab1_repdef
2> with replicate at SRV01_ASE.rdb1
3> go
1> activate subscription prim_tab1_sub for prim_tab1_repdef
2> with replicate at SRV01_ASE.rdb1
3> go
1> validate subscription prim_tab1_sub for prim_tab1_repdef
2> with replicate at SRV01_ASE.rdb1
3> go
1> check subscription prim_tab1_sub for prim_tab1_repdef
2> with replicate at SRV01_ASE.rdb1
3> go

```

Alter repdef

```

=====
** This also fixes the subscription automatically
alter replication definition prim_tab1_repdef
add c char(10) null

```

Testing

```

=====
declare @cnt int
declare @b_val char(10)
declare @c_val char(10)
select @cnt=2
while @cnt<10
BEGIN
    select @b_val='test' + convert(char(5), @cnt)
    select @c_val='test' + convert(char(5), @cnt+@cnt)
    INSERT INTO pdb1..prim_tab1(a,b,c) values (@cnt, @b_val, @c_val)
    select @cnt=@cnt+1
END

```

Warm Standby Install

Setting up warm standby using rs_init can be a bit tricky, so follow these steps below. Watch out for issues with the 'maint' user.

```
=====
In RS
-----
1> create logical connection to "logical_srv"."logical_db"
2> go
In ASE (source)
-----
1> use warmsby
2> go
1> sp_reptostandby warmsby, 'all'
2> go
In ASE (target)
-----
1> use warmsby_copy
2> go
1> sp_reptostandby warmsby_copy, 'all'
2> go
-----

Logins
-----
sp_addlogin warmsby_maint, thisisapassword
go
sp_role 'grant', replication_role, warmsby_maint
go
USE warmsby
go
sp_addalias 'warmsby_maint','dbo'
go
Sync syslogins & sysloginroles (make sure that warmsby_maint is on both ASE servers)
* BCP OUT/IN syslogins between servers

create connection to "SRV1"."warmsby"
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username "warmsby_maint"
set password "thisisapassword"
with log transfer on as active for "logical_srv"."logical_db"

create connection to "SRV2"."warmsby_copy"
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username "warmsby_maint"
set password "thisisapassword"
with log transfer on as standby for "logical_srv"."logical_db" use dump marker

Configure the database for replication:
-----
In PDB: Run in $SYBASE/$SYBASE_REP/scripts/rs_install_primary.sql
isql -SSRV1 -Usa -P<pwd> -Dwarmsby -i rs_install_primary.sql

Configure rep agent
-----
In PDB:
use warmsby
go
sp_stop_rep_agent warmsby
go
sp_config_rep_agent warmsby, 'disable'
go
sp_config_rep_agent warmsby, 'enable', 'repserver', 'repserver_ra', 'repserver_ra_ps'
go
sp_config_rep_agent warmsby, 'priority', '5'
go
sp_config_rep_agent warmsby, 'send buffer size', '16k'
go
sp_config_rep_agent warmsby, 'scan batch size', '1000'
go
sp_config_rep_agent warmsby, 'send warm standby xacts', true
go
sp_start_rep_agent warmsby
go
```

```

sp_setreplicate rs_marker,"true"
go
sp_setreplicate rs_update_lastcommit,"true"
go

Dump'n'Load databases
-----
Immediately dump and load the database from Active to Standby database.
Make sure the "warmsby_maint" has SELECT, DELETE, etc permissions are set on Standby
database
or
use warsby_copy
go
sp_dropuser 'warmsby_maint'
go
sp_addalias 'warmsby_maint', 'dbo'
go

In RS
-----
resume connection to SRV2.warmsby_copy
go

```

Warm Standby Switch over

```

In PDB (old active database)
-----
sp_stop_rep_agent warsby
go

In RS
-----
isql -Uuser [-Syourrepserver]
-To switch over to warm standby server..
admin logical_status
go
--switch active for <logialserver.logicaldb> to <wsserver.wsdbs>
switch active for logical_srv.logical_db to SRV2.warmsby_copy
go
admin logical_status
go

In RDB (New active database)
-----
sp_configure 'enable rep agent threads', 1 -- if not already set
go
sp_start_rep_agent warsby_copy
go

In RS
-----
resume connection to SRV2.warmsby_copy
go

```

Note: if the old primary database has been shutdown or is no longer contactable, the logical status for it will remain as "Suspended/waiting for Enable Marker" until it is fixed. Once the server comes backon line, resume the connection and 'Operation in Progress' will go back to 'None'

Database (MSA) repdef

- * Set up Replication server as normal using rs_init
- * Add primary database to RS using rs_init
- * make sure 'ddl in tran is set on both databases'

```

In PDB
=====
sp_reptostandby $DBNAME,"all"
sp_config_rep_agent pdb1, 'send warm standby xacts', 'true'

In RS
=====
1> create database replication definition pdb1_dbrepdef
2> with primary at SRV1_ASE.pdb1
3> replicate ddl
4> replicate functions

```

```

5> replicate system procedures
6> go

1> create connection to "SRV1_ASE"."test_rep_db"
2> set error class to rs_sqlserver_error_class
3> set function string class to rs_sqlserver_function_class
4> set username to "rep_maint"
5> set password to "rep_maint_ps"
6> go

1> define subscription pdb1_sub
2> for database replication definition pdb1_dbrepdef
3> with primary at SRV1_ASE.pdb1
4> with replicate at SRV1_ASE.test_rep_db
5> subscribe to truncate table
6> use dump marker
7> go

In RDB
=====
To avoid any permission issues in replicate DB
Use test_rep_db
go
sp_addalias 'test_rep_db_maint','dbo'
go

```

At this point the live database should be dumped and loaded into replicate database.
When the dumps have completed, resume the connection to the standby sites.

```

In PRS
=====
resume connection to SRV1_ASE.test_rep_db
go

```

Manually set up connections

```

=====
create connection to server1.dbname
set error class to custom_error_class
set function string class to rs_sqlserver_function_class
set username to dbname_maint
set password to thisisapassword
GO
create connection to server2.dbname
set error class to custom_error_class
set function string class to rs_sqlserver_function_class
set username to dbname_maint
set password to thisisapassword
GO
create connection to server3.dbname_copy2
set error class to custom_error_class
set function string class to rs_sqlserver_function_class
set username to dbname_maint
set password to thisisapassword
GO
alter connection to server1.dbname
set log transfer on
GO
alter connection to server2.dbname
set log transfer off
GO
alter connection to server3.dbname_copy2
set log transfer off
GO

```

setup primary db's for rep

```

=====
for SRV in server1
do
isql -Usa -P<password> -S$SRV -D$DBNAME <
$SYBASE/$SYBASE_REP/scripts/rs_install_primary.sql

```



```

isql -Usa -P<password> -S$SRV <<EOF
exec sp_addlogin dbname_maint,thisisapassword
go
use $DBNAME
go
sp_addalias dbname_maint,dbo
go
exec sp_reptostandby $DBNAME,"all"
go
exec sp_config_rep_agent $DBNAME,enable,repserver_rs,repserver_rs_ra,repserver_rs_ra_ps
go
exec sp_config_rep_agent $DBNAME,"send warm standby xacts",true
go
exec sp_config_rep_agent $DBNAME,'priority','4'
go
exec sp_config_rep_agent $DBNAME,"scan_batch_size","10000"
go
exec sp_config_rep_agent $DBNAME,"send_buffer_size","16K"
go
exec sp_config_rep_agent $DBNAME,"send_structured_oqids","true"
go
exec sp_config_rep_agent $DBNAME,"short_ltl_keywords","true"
go
sp_start_rep_agent $DBNAME
go
EOF
done

```

Function Repdefs (stored procedure replication)

Implementing stored procedure replication by example

In PDB & RDB

```

=====
create table testtable1 (name varchar(10), phone int)
go
create procedure sp__testtable1_insert @name varchar(10), @phone int
as
begin
    insert into testtable1 (name, phone) values (@name, @phone)
end
go

```

```

-- mark sp for replication (ignore error #9137 if using warm stby)
sp_setrepproc sp__testtable1_insert, function
go

```

```

--If your maint user is not dbo in the replicate db, then execute this in the RDB
grant execute on sp__testtable1_insert to maint_user

```

RS

==

Applied function = sp is executed by maint user
Request function = sp is executed by same user who executed SP at the primary database
"create function replication definition" deprecated in repserver 15, use 'applied or requested' instead.

```

-- Note the repdef name exactly matches the proc name.
create applied function replication definition sp__testtable1_insert_repdef
with primary at <logical_srv>.<logical_db>
with all functions named 'sp__testtable1_insert'
(@name varchar(10), @phone int)
go

```

```

-- create subscription
create subscription sp__testtable1_insert_sub
for sp__testtable1_insert_repdef
with replicate at SRV2_ASE.test_rep_db
without materialization
go
check subscription sp__testtable1_insert_sub
for sp__testtable1_insert_repdef
with replicate at SRV2_ASE.test_rep_db
go

```

```
-- TESTING in PDB--
sp__testtable1_insert 'gary', 1234
go

--Dropping a function definition
drop function replication definition sp__testtable1_insert_repdef
```

Replication Tuning Notes

=====

Golden rules

=====

1. Never have repdefs, which are not subscribed to. All transactions on replicated tables are sent to the Inbound Queue (IBQ), sorted into commit order and translated to Log Transfer Language (LTL). Only then are they checked for subscriptions. This results in wasted space in the IBQ and processing by the SQT manager.
2. Make sure SQT has enough memory allocated. Also, check memory_limit
rs_configure 'sqt_max_cache_size' to 'xxxxx'

Find Bottlenecks

=====

```
select * from master..syslogshold  --check for large uncommitted transactions.
```

Measure diff between repagent position and end of log (1TP & 2TP)

```
-----
--rep agent - value of 'Current Marker' column, example (53550,1)
sp_help_rep_agent <db_name>
-- read until end of log
dbcc traceon(3604)
dbcc pglinkage(<dbid>, <current_marker>, 0,2,0,1)
example: dbcc pglinkage(5, 53550, 0,2,0,1)
example output: "3909 pages scanned"
-- So repagent if 3909 pages behind log truncation marker.
-- We should have very little lag!
(see rs_checklag.ksh in
```

Measure IBQ & OBQ size

```
-----
admin who, sqm
Info column of XXX:0 = IBQ
Info column of XXX:1 = OBQ
difference between 'Last Seg.Block' & "Next Read" should be minimal
Example: Last Seg.Block = 226.64
         Next Read = 140.50.13
         (226-140) = 86 Mb in IBQ
```

This info is also stored in the RSSD db in rs_diskpartitions, rs_segments
This is used in sp__queueinfo (see Appendix A)

Check what is in the queues

Once we know which queues are filling up, use the command below to determine the sql in the queues.

```
sysadmin dump_queue, <q_num>, <q_type>, -1, -2, -1, client
or
sysadmin log_first_tran, <srv>, <dbname>
```

Check ASE activity

If monitoring tables are installed, discover busiest spid and extract SQL.
(Useful tools for this: sp__mon_sql2 & sp__capture_sql)

Sizing caches sizes

SQT_MAX_CACHE_SIZE

To check if this parameter has been set too low, run an "admin who,sqt" at regular intervals and check if the columns "removed" or "full" contain a non-zero value.

Example:-

```
configure replication server set sqt_max_cache_size to '115343360' --bytes (110MB)
```

DSI_SQT_MAX_CACHE_SIZE

"admin who,sqt" also shows the DSI threads. So when "removed" or "full" are non-zero for a DSI thread, change the memory setting at the connection-level using "alter connection to <dataserver.database> set dsi_sqt_max_cache_size to '<new-value>'"

It's best to start increasing "sqt_max_cache_size" before changing settings at the connection level. Do not increase sqt_max_cache_size to much. Oversizing the cache will in fact decrease performance.

Tuning

=====

Tuning Primary DB

```
-----
sp_help_rep_agent <db_name>, 'config'
sp_config_rep_agent <db_name>, scan_batch_size, '10000' --max num records sent to RS
sp_config_rep_agent <db_name>, 'batch_ltl', 'true' --LTL cmds batched up then sent to RS
sp_config_rep_agent <db_name>, send_buffer_size, '16k' -- network packet size
sp_config_rep_agent <db_name>, priority, '2' --default is 5. lower=higher priority
WARNING: making changes to the rep agent can cause a warm stby connection to fail, if the
replicate DB name is different. Requires a resume connection..skip transaction. And the
config changes to be repeated at the replicates rep agent.
```

Tuning Rep Server

```
-----
Note: 1 Repserver = 1 CPU
admin who, sqt
existing values are stored in RSSD. Use:
select optionname, charvalue from rs_config
configure replication server set sqt_max_cache_size to '20971520' --in RS, or
rs_configure 'sqt_max_cache_size' to 'xxxxx' -- in RSSD,
Ensure value of (sqt_max_cache_size * num. of queues) is less than memory_limit.
Suggest setting sqt_max_cache_size to 20mb (20971520 bytes)
Max memory_limit = 2047 (just under 2Gb)
Use RAW device for Stable Device.
rs_configure 'num_threads', 75 -- if using Open Server (replicating to non Sybase DB)
configure replication server set sqm_write_flush to 'dio' -- only on Unix file systems
```

Tuning RSSD

```
-----
sp_config_rep_agent <db_name>, priority, '2' --RSSD can have it's own repagent
Put on same machine as RS.
use 'localhost <port>' in interfaces file for ASE and RS
example:
REPL_RS
```

```
master tcp ether localhost 10010
master tcp ether <server> 10010
query tcp ether <server> 10010
```

```
--keeps rs system tables in memory.
configure replication server set sts_full_cache_rs_classes to 'on'
configure replication server set sts_full_cache_rs_columns to 'on'
configure replication server set sts_full_cache_rs_config to 'on'
configure replication server set sts_full_cache_rs_databases to 'on'
configure replication server set sts_full_cache_rs_datatype to 'on'
configure replication server set sts_full_cache_rs_diskaffinity to 'on'
configure replication server set sts_full_cache_rs_functions to 'on'
configure replication server set sts_full_cache_rs_objects to 'on'
```

```

configure replication server set sts_full_cache_rs_publications to 'on'
configure replication server set sts_full_cache_rs_queues to 'on'
configure replication server set sts_full_cache_rs_repdb to 'on'
configure replication server set sts_full_cache_rs_routes to 'on'
configure replication server set sts_full_cache_rs_sites to 'on'
configure replication server set sts_full_cache_rs_systext to 'on'
configure replication server set sts_full_cache_rs_translation to 'on'
configure replication server set sts_full_cache_rs_users to 'on'
configure replication server set sts_full_cache_rs_version to 'on'
*note: in repserver 15.0, do not cache rs_locator. repserver crash can cause
inconsistencies.

```

Tuning Replicate DB

```

-----
change maint user priority in ASE
drop referential integrity checks (foreign keys)
use func. strings instead of triggers.

```

Tuning DSI

```

-----
Increase replicate-ASE no. of locks
dsi_max_xacts_in_group
alter connection to RDS.rdb set db_packet_size to 'xxx'
switch on replicate minimal columns --use all columns if replicating to non-Sybase DB
Use parallel DSI threads (do not do this lightly):-
parallel_dsi (sets standard values on multiple settings below)
  dsi_num_threads
  dsi_serialization_method
{none|wait_for_commit|isolation_level_3|single_transaction_per_origin}
  dsi_sq_max_cache_size
  dsi_large_xact_size
  dsi_num_large_xact_threads
  dsi_partitioning_rule

** Recommend using dsi_serialization_method 'none' followed by 'isolation_level_3'
** Recommend using 'time' partitioning

```

Configure for SMP systems

```

-----
When replication server runs on a multiprocessor machine you can take advantage of the multi-
threaded capability of RepServer. Run this command:-
  configure replication server set smp_enable to 'on'

```

Minimal Column Replication

Repserver wants to replicate all columns for a modified row, even the values that have not been changed.
Example:

```

UPDATE customer SET firstname="joe" where ID=202
Results in this getting replicated
UPDATE customer SET firstname="joe", surname='blogs', address="22 archia ave",
tel="020 123 4567" where ID=202

```

but with minimal column replication set on only the 'firstname' data change is replicated. This reduces the queue size, processing time of the repserver and increases throughput.

To enable define 'replicate_minimal_replication' in your table repdef or at the connection level.

For more tuning advice, see
http://www.petersap.nl/Sybasewiki/index.php?title=Performance_Tuning&printable=yes

Monitor Counters

=====

Not requiring Setup

```
rs_helpcounter (ref's table rs_statcounters)
admin statistics, SQM, ByteSize
admin statistics, reset
admin statistics, sysmon "00:00:10"
```

Requiring Setup

```
select * from rs_statdetails, rs_statrun
setup:
set stat_sampling to 'on'
admin stats_intrusive_counter, 'on'
stats_flush_rssd to on
stat_reset_afterflush to on
stat_daemon_sleep_time to '600'
admin stat_config_module, 'all_modules', 'on'
admin stat_config_connections
admin statstatistics, flush_statistics
See white paper: "Sybase Replication Performance and Tuning" by Jeff Tallman
http://my.sybase.com/detail?id=1015811
```

Disaster Recovery Notes

=====

Recover from reloading Primary Database

```
once loaded and online,
## on pdb:
dbcc settrunc(ltm, ignore)
--move log trunc marker (LTP) to new page
create table dummy_table (a char(255), b char(255))
go
insert dummy_table values ('a', 'b')
go 40
drop table dummy_table
go
dump transaction <pdb> with truncate_only
go
--re-establish 2TP:
dbcc settrunc(ltm, valid)
go
--Now set it to zero
use <rssd>
go
rs_zero_ltm <ase>, <pdb>
go
## on RS:
admin get_generation, <ase>, <pdb>
## on pdb:
--update generation no. (new number >= old number)
--however, setting to 0, is normally ok
dbcc settrunc(ltm, gen_id, <new number>)
#-----#
# now resync data with replicate db #
#-----#
#on RS:
resume connection to <ase>.<pdb>
# on pdb:
```

```
exec sp_start_rep_agent <pdb>
```

****If only a few tables are out of sync, you can use Sybase command-line utility called rs_subcmp**

Skipping transactions

```
-----
--If we encounter a duplicate insert error
#on RS:
resume connection to <ase>.<rdb> skip transaction
#on RSSD:
--find transaction id
rs_helpexception
--get SQL
rs_helpexception <tran_id>, v
```

Stop Replication

```
-----
#on pdb:
select * from master..syslogshold where dbid=db_id(<pdb>)
go
sp_stop_rep_agent <pdb>
go
dbcc settrunc(1tm, ignore)
go
```

Replaying Transaction Logs

```
-----
restart RS in single user mode (-M switch)
#on RS:
set log recovery for <ase>.<pdb>
allow connections
go
-- Method shows the use of temporary database to hold database.
create database called 'temp_rep' then configure for replication.
use temp_rep
go
exec sp_config_rep_agent temp_rep, 'enable', '<RS>', 'sa', '<passwd>'
go
use master
go
load database temp_rep from '<dump_file>'
go
-- the "connect database" refers to <pdb>
exec sp_start_rep_agent temp_rep, recovery, '<ase>', '<pdb>', '<RS>'
go
--Once complete, RepAgent will shutdown
--Now repeat these steps for each tran. log. Load and start RepAgent.
--** Check replication Server errorlog for any messages about "loss detection". If none
found...
--restart RS in normal mode.
#on pdb
--put back 2TP
dbcc settrunc(1tm, valid)
go
sp_start_rep_agent <pdb>
go
--drop temp_rep!
```

Rebuild a Stable Device - with tran log

```
-----
If all threads are down, it may be because the Stable Device is corrupt or missing.
Check for OFFLINE disk partitions
#on RS:
admin disk_space
go
Oh dear! the transactions on the SD IBQ have gone but fear not, they are still in the
transaction logs.
on the RS, threads are DOWN, resume and suspended connecions. Should still show DOWN.
```

```

#on RS:
drop partition <partition name>
go
--In Unix touch the new file
add partition <part_name> on '<physical_name>' with size <size in Mb>
go
admin disp_space
go
--You should see old disk as DROPPED and your new disk ONLINE
--Now rebuild SD from the transaction logs
rebuild queues
go
resume connection to <ase>.<rdb>
--Check RS errorlog and wait for msg "Rebuild Queues: Complete" & "DSI: detecting loss
for dataserver <ase>.<rdb>"
#on pdb:
exec sp_start_rep_agent pdb
go
--Check RS errorlog for "loss detection" messages. If none found, normal replication will
continue.
--You can force replication to continue using
#on RS:
ignore loss from <ase>.<pdb> to <ase>.<rdb>
go
#on RS:
--Old partition should have disappeared from server. You can drop file/device.
admin disk_space

```

Rebuild a Stable Device - without tran log

Once again the SD has disappeared but this time the transaction log has been TRUNCATED.
#on the RS, threads are DOWN, resume the suspended connections. Should still show DOWN.

Repeat steps for "Rebuild a Stable Device - with tran log" but this time ou will see
"loss detection" in the RS errorlog. This time we must ignore the loss.

```

#on RS:
ignore loss from <ase>.<pdb> to <ase>.<rdb>
go
#-----#
# now resync data with replicate db #
#-----#

```

Restore the RSSD from backup

```

-----
#on pdb:
sp_stop_rep_agent pdb
go
#on RS:
shutdown
go
--restore RSSD from backup. Once complete, proceed
--If RSSD had rep agent, start RS (or skip to --else)
#on RSSD:
dbcc settrunc(ltm, valid)
go
#on RS:
admin get_generation, <ase>, <rssd_db>
go
shutdown
--else,
restart RS in single user mode (-M)
#on RS:
resume connection to <ase>.<rdb>
go
rebuild queues
go
#on pdb:
exec sp_start_rep_agent <pdb>, recovery
go
--** Check replication Server errorlog for any messages about "loss detection". Hoefully,
you have none.

```

General Troubleshooting

Stable Queue Full

```
Double check queue is full
In RSSD
=====
rs_helppartition

restart rep agent and connections
=====
In PDB
-----
sp_help_rep_agent pdb
sp_stop_rep_agent pdb
sp_start_rep_agent pdb (status should be not active)

In RS
-----
Suspend connection to server1.pdb
Resume connection to pdb

Increase stable queue
=====
In RS
-----
admin disk_space (shows existing partitions)

touch /usr/replication/queue10.dat

add partition sq_part10 on '/usr/replication/queue10.dat' with size 1000 (in Mb)

You can use "drop partition sq_part10" online at a later time
```

Ignoring duplicate keys – when we have a lot, use error class!

Sybase's Replication Server allows you to replicate data entry from one database into another (there can be more than one replicate database). They don't necessarily have to be even from the same vendor. Duplicate rows will occur when an application inserts data into the primary and replicate database(s), if the data being entered in a replicated table. Replication Server's DSI connection will stop saying that it has detected a duplicate key and requires a DBA to tell it what to do. If this duplicate key can be ignored, then the DBA will skip the transaction, which will make a note of the transaction and will skip it (go on to the next transaction).

```
1: REP_SERVER> resume connection to MYSERVER.MYDB skip transaction
2: REP_SERVER> go
```

The problem with this approach is that if there are a lot of duplicate keys, not only could you be sitting for a while skipping the transactions, you run the risk of skipping a transaction that isn't a duplicate key. Say if someone deleted the table on the replicate database.. You could easily make a mess of things if you arbitrarily skip transactions.

Replication Server has a feature called error classes that you can define the course of action if an error occurs with a DSI connection. The only real issue is that the lowest level of granularity is at the DSI connection level and the highest is all insert dbms type (i.e. ASE) replicated systems. To create an error class:

```
1: REP_SERVER> create error class ASEallowdupsErrorClass
2: REP_SERVER> go
```

The error classes can be inherited so if you wanted an error class to ignore duplicate keys and another to stop replication on a duplicate key, you would do something like so:

```
1: RSSD> rs_init_erroractions ASEallowdupsErrorClass, rs_sqlserver_error_class
2: RSSD> go
```

Sybase ASE's error number for a duplicate key is 2601, but ASE will also raise the 3621 (aborted transaction) error. We need to set the error class ASEallowdupsErrorClass to ignore duplicate keys:

```
1: REP_SERVER> assign action ignore for ASEallowdupsErrorClass to 2601
2: REP_SERVER> go
1: REP_SERVER> assign action ignore for ASEallowdupsErrorClass to 3621
```


2: REP_SERVER> go

Now that we've created the error class and set it to ignore duplicates, we need to do two last things:

alter the DSI connections to use the new error class
suspend and then resume the DSI connections for the DSIs to use the new error class

```
1: REP_SERVER> alter connection to MYSERVER.MYDB
2: REP_SERVER> set error class to ASEallowdupsErrorClass
3: REP_SERVER> go
1: REP_SERVER> suspend connection to MYSERVER.MYDB
2: REP_SERVER> go
1: REP_SERVER> resume connection to MYSERVER.MYDB
2: REP_SERVER> go
```

Generally, applications should not be performing data entry of the same data across the replicated databases as Replication Server is made for it.

Reverse Engineering an Error Class

Sometimes we want to recreate an existing error class, for example, taking one from production into a new UAT environment. It is not really possible to do this but we can work out the modified error codes in the user defined error class and then manually recreate the class. Run the SQL, in the RSSD, below and pipe the output into 2 files

First file

```
=====
select ds_errorid, action=v.name
from rs_erroractions e, rs_classes c, rs_tvalues v
where e.errorclassid=c.classid
and e.action=v.value
and v.type='ERR'
and c.classname='rs_sqlserver_error_class'
order by 1
go
```

Second File

```
=====
select ds_errorid, action=v.name
from rs_erroractions e, rs_classes c, rs_tvalues v
where e.errorclassid=c.classid
and e.action=v.value
and v.type='ERR'
and c.classname='ASEallowdupsErrorClass'
order by 1
go
```

Now do a 'diff' against these files and any different codes will be displayed. To find out what the codes are, in RSSD

rs_helperror 2601, 'v'

HowTo determine the error class configured for a connection

To determine the error class configured for a connection, run this query in the RSSD:

```
select dsname, dbname, classname 'Error class'
from rs_databases d, rs_classes c
where d.errorclassid = c.classid
```

Row count mismatch – use a replication server error class. **V15.2+**

You may get this error in the errorlog:

"Row count mismatch for the command executed on 'dataserver.database'. The command impacted x rows but it should impact y rows."


```
go
Sp_config_rep_agent, <pdb>, 'disable' --check master..syslogshold to confirm
go
```

In RSSD

```
=====
delete from rs_subscriptions where subname='<subname>'
go
delete from rs_dbreps where dbrepname='<db_repdef_name>'
go
```

Now you can drop connections!

Detecting loss

Sometimes replication stops without an error. This could happen after a restore of the primary database. If message loss occurs we will not always see this using 'admin who' and repserver might not print a 'detecting loss' message to the errorlog. Check the rs_oqid and rs_exceptslast in the RSSD and to see if some of the queues show a status of '2' which indicates that the queue is suspended due to lost messages.

If repserver has not correctly recognised that loss has occurred, then in order for repserver to ignore these errors, we must get it to 'find' them. Restart repserver and check the errorlog for message:

```
DSI: detecting loss for database
In RS
=====
Ignore loss from prim_server.prim_db
go
```

Repserver Trace Flags

The following Rep Server traceflags will track the commands being written to the stable queue, and being passed to the Replicate dataserver.

Flag: SQM, SQM_TRACE_COMMANDS

This flag is used when you want to know what commands have been written to the stable queue.

Flag: DSI, DSI_BUF_DUMP

Use this flag when you want to know what is in the language command buffer passed to dbcmd()

Replication Server accepts on-line trace command from isql as follows:

```
trace { "on" | "off" }, module, trace_flag
```

e.g., trace "on",sqm,sqm_trace_commands

both module and trace flag can be either upper or lower case.

Replication Server accepts trace flags from the config file. The syntax is
trace=module,trace_flag

e.g., trace "on", dsi,dsi_buf_dump

keep in mind that these will trace ALL commands, so will produce large amounts of output.

Configure the rep agent to trace LTL--write output to a trace file (not to ASE log)

```
isql -Uxx -Pxx -SActive_Server
>use PDB
>go
>sp_stop_rep_agent PDB
>go
```

Turn on Rep Agent tracing and DSI/function string tracing

(in the following command, supply full path and filename for trace filename--trace_log_file is required and must be enclosed in double quotes"
 >sp_config_rep_agent PDB, "trace_log_file", "<trace_filename>"
 >go
 >sp_config_rep_agent PDB, "traceon", "9201"
 >go
 >sp_start_rep_agent PDB
 >go

When the Rep Agent appears to stop responding, collect

```
sp_who
go

get spid of RA

dbcc pss
dbcc stacktrace (<spid>)
```

Turn off Rep Agent tracing and DSI/function string tracing

```
>sp_stop_rep_agent
>go
(to disable, replace the trace file name with "")
>sp_config_rep_agent <dbname>, "trace_log_file", ""
>go
>sp_config_rep_agent <dbname>, "traceoff", "9201"
>go
>sp_start_rep_agent
>go
```

Handy Tips

Renaming a replicate database

Repserver used a ID to track databases but it also holds the database name in 3 tables. rs_databases, rs_repdb, rs_idnames

Run 'select *' against these 3 tables and then depending on the DBID, run the examples below (obviously, change the values)

```
update rs_databases set dbname=" NEW_DatabaseName " where dbid=152
go
update rs_repdb set dbname="NEW_DatabaseName" where dbid=152
go
update rs_idnames set name2="NEW_DatabaseName" where id=152
go
shutdown
go
```

Now rename the ASE database using 'sp_renamedb' (in single user mode)

Restart repserver. The name change should be complete.

rs_ticket Feature

Run this at the primary database and collect information about the 'ticket' progress at the replicates.

Setup

Assuming you have added the databases to the repserver using rs_init, all the necessary objects will be in the database. If you added your databases in a different way, you will need to use either rs_init and execute "Upgrade an existing database in the replication system" or manually run the scripts manually from %Sybase%\REP-15_5\scripts\rs_install_primary.sql (I have not tested the using the scripts method).

Exclude rs_ticket_history from replication

You must exclude the rs_ticket table from replication for MSA and warm standby. Do this by modifying the database repdef

Database Replication - example of exclusion DDL:

```
alter database replication definition <dbname>_dbrepdef
with primary at <servername>.<dbname>
not replicate tables in ('rs_ticket_history', .....<any other excluded tables>)
```

Check by listing the current exclusions in the RSSD

```
1> select convert(char(35), s.name), s.type, convert(char(35), db.dbrepname) from
rs_dbsubsets s, rs_dbreps db where s.dbrepid=db.dbrepid
2> go
name
type dbrepname
```

```
-----
rs_ticket_history
T      <dbname>_dbrepdef
```

To stop the rs_ticket_history getting excluded from **Warm Standby** replication you need to use a different method than MSA. Since repserver 15.2 you can use the 'never' option at the primary database:-

```
sp_setreptable rs_ticket_history, 'never'
'never' – Disables replication on the table, regardless of the database replication setting.
```

Send your first 'ticket'

in the PDB

```
=====
-- Basically, you can put anything into the 4 custom arguments for rs_ticket.
-- we normally use a client ID for the 1st argument and the PDB details in the 4th
-- but you can use anything.
-- The 4th argument can contain more text.
exec rs_ticket "UK-1.2", "", "", "<servername>.<dbname>"
```

Checking the results of your ticket

in the RDB

```
-----
select * from rs_ticket_history
-- or use this to get the last entry
select * from rs_ticket_history where cnt=(select max(cnt) from rs_ticket_history)
```

How to implement

Write a script to insert a 'ticket' at the primary DB by running the rs_ticket command every x minutes. See Appendix A for an example rs_ticket.sh script

Get data from the RDB to trend the latency

The results of the rs_ticket process are stored in a table called rs_ticket_history, at each replicate. Below is a query to see latency for replication to a particular replicate over the last 24 hours – summarised by average and maximum latency by hour (hour relative to when you run the query).

This shows the breakdown of the ticket's journey:

p2r=PDB to RDB (i.e. whole trip)

p2e=PDB to Executor thread on the PRS – where the LTL is received.

e2d=Executor to DIST thread

d2d=DIST to DSI thread

d2r=DSI to RDB

For replication to RDB – Summarise by Hour part I:

In RDB

```

-----
select datediff(hh,pdb_t,getdate()) 'Hours Ago',
avg(datediff(ss,pdb_t,rdb_t)) 'avg p2r', max(datediff(ss,pdb_t,rdb_t)) 'max p2r',
avg(datediff(ss,pdb_t,exec_t)) 'avg p2e', max(datediff(ss,pdb_t,exec_t)) 'max p2e',
avg(datediff(ss,exec_t,dist_t)) 'avg e2d', max(datediff(ss,exec_t,dist_t)) 'max e2d',
avg(datediff(ss,dist_t,dsi_t)) 'avg d2d', max(datediff(ss,dist_t,dsi_t)) 'max d2d',
avg(datediff(ss,dsi_t,rdb_t)) 'avg d2r', max(datediff(ss,dsi_t,rdb_t)) 'max d2r'
from rs_ticket_history
where pdb_t > dateadd( hh, -24, getdate() )
group by datediff(hh,pdb_t,getdate())
order by datediff(hh,pdb_t,getdate()) desc
go

```

Hours Ago	avg p2r	max p2r	avg p2e	max p2e	avg e2d	max e2d	avg d2d	max d2d	avg d2r	max d2r
23	1	3	0	0	0	2	1	1	0	1
22	1	2	0	0	0	2	1	1	0	0
21	1	2	0	0	0	1	1	1	0	0
20	1	3	0	0	0	3	1	1	0	0
19	1	2	0	0	1	2	1	1	0	1
18	1	2	0	0	1	2	1	1	0	1
17	1	2	0	0	1	1	1	1	0	1
16	1	2	-1	-1	0	1	1	1	1	1
15	1	2	0	0	0	1	1	1	0	1
14	1	3	0	0	0	2	1	1	0	1
13	1	3	0	0	1	2	1	1	0	1
12	1	2	0	0	0	2	1	1	0	0
11	1	2	0	0	0	2	1	1	0	1
10	1	3	0	0	0	2	1	2	0	1
9	1	3	0	0	1	2	1	1	0	1
8	1	2	0	0	0	2	1	1	0	1
7	1	3	0	0	0	2	1	1	1	1
6	1	3	0	0	0	2	1	1	0	1
5	1	2	0	0	0	1	1	1	0	1
4	1	2	0	0	1	2	1	1	0	0
3	1	3	0	0	0	2	1	1	0	0
2	1	2	0	0	0	2	1	1	0	1
1	1	3	0	0	0	2	1	1	0	1
0	1	4	0	0	0	3	1	1	0	1

(24 rows affected)

So we had a maximum latency of 4 seconds within the last hour – though clearly by the appearance of some negative numbers the host clocks are not in sync with a high precision.

For replication to RDB – Summarise by Hour part II:

Or you can do something similar but relative to the last hour of the clock e.g. for the last 12 clock hours:

In RDB

```

-----
select datepart(hh,pdb_t),
avg(datediff(ss,pdb_t,rdb_t)) 'avg p2r', max(datediff(ss,pdb_t,rdb_t)) 'max p2r',
avg(datediff(ss,pdb_t,exec_t)) 'avg p2e', max(datediff(ss,pdb_t,exec_t)) 'max p2e',
avg(datediff(ss,exec_t,dist_t)) 'avg e2d', max(datediff(ss,exec_t,dist_t)) 'max e2d',
avg(datediff(ss,dist_t,dsi_t)) 'avg d2d', max(datediff(ss,dist_t,dsi_t)) 'max d2d',
avg(datediff(ss,dsi_t,rdb_t)) 'avg d2r', max(datediff(ss,dsi_t,rdb_t)) 'max d2r'
from rs_ticket_history
where pdb_t > dateadd( hh, -12, getdate() )

```

```
group by datepart(hh,pdb_t)
order by datepart(hh,pdb_t)
go
```

	avg p2r	max p2r	avg p2e	max p2e	avg e2d	max e2d	avg d2d	max d2d	avg d2r	max d2r
0	1	1	3	0	0	0	2	1	2	0
1	1	1	2	0	0	1	2	1	1	0
2	1	1	2	0	0	1	1	1	1	0
3	1	1	2	0	0	0	2	1	1	0
4	1	1	3	0	0	0	2	1	1	0
5	1	1	2	0	0	0	1	1	2	0
6	1	1	2	0	0	0	1	1	2	0
7	1	1	3	0	0	0	2	1	1	0
8	1	1	2	0	0	0	2	1	1	0
9	1	1	2	0	0	0	1	1	1	0
10	1	1	4	0	0	1	3	1	1	0
22	1	1	1	0	0	1	1	1	1	0
23	1	1	2	0	0	0	2	1	1	0

(13 rows affected)

The red row is for 10am-11am, 22 and 23 are for last night.

Apart from time there is also information in the tickets for commands & bytes processed at various points through the RS – it should be possible to get meaningful information on replication load/throughput by computing successive differences between these and then to put this against the latency stats.

From replication to RDB – Recent Latency Report:

Here is an example of latency and data sizes for every ticket (we submitted a ticket every minute).

Change the value of \$PERIOD to the number of minutes you wish to go back

```
select cnt, pdb_t 'Time at Primary', convert(varchar(30), rdb) 'Replicate DB', datediff(ss,pdb_t,rdb_t) 'p2r Ltny (sec)',
convert(int, (a.exec_b - (SELECT b.exec_b FROM rs_ticket_history b WHERE b.cnt = (a.cnt - 1)))/1024, 0) 'KBytRecExc',
convert(int, (a.rsi_b - (SELECT b.rsi_b FROM rs_ticket_history b WHERE b.cnt = (a.cnt - 1)))/1024, 0) 'KBytRecRSI',
convert(int, a.dsi_tnx - (SELECT b.dsi_tnx FROM rs_ticket_history b WHERE b.cnt = (a.cnt - 1)), 0) 'DSINumTrns',
convert(int, a.dsi_cmd - (SELECT b.dsi_cmd FROM rs_ticket_history b WHERE b.cnt = (a.cnt - 1)), 0) 'DSINumCmds',
from rs_ticket_history a
where pdb_t > dateadd(mi, -$PERIOD, getdate())
order by pdb_t
go
```

cnt	Time at Primary	Replicate DB	p2r Ltny (sec)	KBytRecExc	KBytRecRSI	NumTrns	NumCmds
60468	Jan 23 2014 2:46PM	NOVIA_UAT_CW_RO_Composer	2	120	0	130	393
60469	Jan 23 2014 2:47PM	NOVIA_UAT_CW_RO_Composer	2	120	0	131	390
60470	Jan 23 2014 2:48PM	NOVIA_UAT_CW_RO_Composer	2	119	0	129	387
60471	Jan 23 2014 2:49PM	NOVIA_UAT_CW_RO_Composer	2	120	0	130	393
60472	Jan 23 2014 2:50PM	NOVIA_UAT_CW_RO_Composer	2	129	0	135	420
60473	Jan 23 2014 2:51PM	NOVIA_UAT_CW_RO_Composer	2	110	0	125	357
60474	Jan 23 2014 2:52PM	NOVIA_UAT_CW_RO_Composer	2	120	0	130	390
60475	Jan 23 2014 2:53PM	NOVIA_UAT_CW_RO_Composer	2	121	0	131	393
60476	Jan 23 2014 2:54PM	NOVIA_UAT_CW_RO_Composer	2	119	0	129	387
60477	Jan 23 2014 2:55PM	NOVIA_UAT_CW_RO_Composer	2	121	0	130	393

Appendix A – Shell scripts

rs_checkreplag.ksh

```
#!/bin/ksh
#####
#
# work out the lag in Mb between 1TP & 2TP markers in a replicated database
# By G. Devine
#
#####
if [ $# -ne 3 ]
then
    echo Usage: $(basename $0) LOCAL_SERVER TARGET_SERVER DBNAME
    exit 1
else
    LOCALSRV=$1
    TRGSRV=$2
    DBNAME=$3
fi
. /opt/home/sybase/admin/.syb_cfg.sh $LOCALSRV
USERNAME=sa
PWD=`grep ${LOCALSRV}, /opt/home/sybase/admin/.servers | awk -F',' '{print $4}'`
OUTFILE= `basename $0 .out.$$`
#----- MAIN -----#
# Get the Current Marker for the rep agent
isql -U$USERNAME -S$TRGSRV -D$DBNAME -w1024 <<-EOF | egrep -v "Password:|return status" |
sed -e '1,3d' > $OUTFILE
    $PWD
    --set nocount on
    sp_help_rep_agent ${DBNAME}, scan
    go
    EOF

CURRMARKER=`cat $OUTFILE | awk '{print $4}' | sed -e 's/(//g' -e 's/)//g' | awk -F',' '{print $1}'`
rm $OUTFILE

# Now work out the pages scanned between 1TP & 2TP
isql -U$USERNAME -S$TRGSRV -D$DBNAME -w1024 <<-EOF | egrep -v "Password:|return status" >
$OUTFILE
    $PWD
    set nocount on
    dbcc traceon (3604)
    go
    declare @dbid_num int
    select @dbid_num=db_id('${DBNAME}')
    dbcc pglinkage(@dbid_num, $CURRMARKER, 0,2,0,1)
    go
    EOF
PAGESCANS=`cat $OUTFILE | grep 'pages scanned' | awk '{print $1}'`
rm $OUTFILE

# Determine server page size
isql -U$USERNAME -S$TRGSRV -D$DBNAME -w1024 <<-EOF | egrep -v "Password:|return status" >
$OUTFILE
    $PWD
    select 'ABCDEFG' + convert (varchar (7), @@maxpagesize) + 'ABCDEFG'
    go
    EOF

PAGESIZE=`cat $OUTFILE | grep 'ABCDEFG' | sed -e 's/ABCDEFG//g'`
rm $OUTFILE

### Do the calculations ###
BYTESCAN=`expr $PAGESCANS \* $PAGESIZE`
LAGSCAN=$(echo "scale=5; $BYTESCAN / 1024 / 1024" | bc)

### Result ###
echo
echo " Difference between 1TP & 2TP for database $DBNAME is $LAGSCAN MB"
```


sp__queueinfo

Use admin rssd_name to get the name of the RSSD database, if you do not already know it. Then substitute into the code below.

```
create proc sp__queueinfo
as
set nocount on
declare @total varchar(10),
        @free varchar(10),
        @freeperc varchar(10),
        @repserver varchar(30),
        @datetime varchar(20)

select @repserver = charvalue from <rssd_dbname>..rs_config where optionname = 'oserver'

select @datetime = convert(varchar(10),getdate(),101)+" "+convert(varchar(8),getdate(),8),
       @total = convert(varchar(10),sum(num_segs)),
       @free = convert(varchar(10),sum(num_segs)-sum(allocated_segs)),
       @freeperc = convert(varchar(12),convert(numeric(10,2),
       (convert(real,(sum(num_segs)-sum(allocated_segs))) /
convert(real,sum(num_segs))*100 ))

from <rssd_dbname>..rs_diskpartitions

print "Stable Queue Information for %1! at %2!",@repserver, @datetime
print "Total Partition Size = %1!MB, Space Remaining = %2!MB
(%3!%)",@total,@free,@freeperc

select rs.q_number,rs.q_type, ( select dsname+'.'+dbname
                             from <rssd_dbname>..rs_databases
                             where dbid = rs.q_number
                             and rs.q_number != 0) queue_name, count(*) "size(MB)"
from <rssd_dbname>..rs_segments rs
group by q_number,q_type
having q_number != 0
order by count(*) desc
```

rs_ticket.sh

```
#!/bin/sh
# Description:
# Runs rs_ticket on the PDS.PDB's specified as arguments.
# Intended to be called from crontab on a regular basis.
# History:
# Date           who           Issue           Description
# -----
# 01/11/2013     Alex Vickers   N/A           Initial Writing.
# 12/12/2013     Garrett Devine  1.1          add fnDisplayUsage & minor
changes
set +x
set +e
set +u

#source environment
. /home/dds/.sybenv local

USERNAME=sa
SCRIPTDIR=/home/dds/scripts
SYBASE=/opt/sybase_15_5_0_esd3
SYBASE_OCS=OCS-15_0
export SYBASE SYBASE_OCS
ISQL=${SYBASE}/${SYBASE_OCS}/bin/isql
#####
#####
# FUNCTIONS
#####
#####
fnDisplayUsage()
{
set +x
# function: Displays usage of command
#
```

```

echo " Usage: `basename $0` <set of pdb/rdb pairs"
echo
echo " Examples:"
echo
echo "      ./`basename $0` AGUAT_DB_CMP.AG_UAT_Composer AGUAT_DB_CW.AG_UAT_ComposerWeb"
}
#####
#####
# Check if anything has been past at the command line
if [ $# -le 1 ]
then
    fnDisplayUsage
fi

for arg in $*; do
    SRV=$(echo $arg | cut -f1 -d.)
    DB=$(echo $arg | cut -f2 -d.)
    PASSWORD=$(grep "^$SRV,sa," ${SCRIPTDIR}/servers.txt.sa | /bin/awk 'BEGIN{FS=","};{print $3};')
    echo "Running for $SRV.$DB"
    $ISQL -U$USERNAME -S$SRV -D$DB -P$PASSWORD <<-EOD
    set nocount on
    set proc_return_status off
    if db_name() = "$DB"
    begin
        declare @arg4 varchar(50)
        select @arg4=@@servername+'.'+db_name()
        exec rs_ticket 'UK-1.2', "", "", @arg4
    end
go
EOD
Done

Example crontab entry
0-59 * * * * /home/bin/rs_ticket.sh <prim_srv1>.<prim_db1> <prim_srv2>.<prim_db2>
>/home/log/rs_ticket.sh.UAT.log 2>&1

```

Appendix B – troubleshooting

Uninstall repserver program

if you want to trash your repserver and start over again, you may find that it will not uninstall. If that is the case, follow these instructions

The installer reads and maintains version information in a file called "vpd.properties", which is probably still located in the "C:\Windows" directory; removing the install directory of repserver won't remove this file.

Please do the following:

1. rename the vpd.properties file at C:\windows or the drive where your Windows is installed
2. go into Control Panel, create a new system environment variable "INSTALL_ALL_PATCH", and give it any value (e.g. "1")
3. install the repserver
4. remove the "INSTALL_ALL_PATCH" variable

Logical Connection will not Drop

If you get an error like the following
 "1> drop logical connection to COMPOSER_DS.SIT_Composer
 2> go
 Msg 15236, Level 12, State 0:
 Server 'AGSIT_DB_REP_RS':

Can not drop logical connection to COMPOSER_DS.SIT_Composer because either subscriptions of repdefs exist for it"

Check
 select * from rs_databases
 select * from rs_object

if the rs_databases.. dist_status or src_status are greater than 1, then this indicates an issue. The connection could have any of the following

Status of the connection. Can be:

- 0x1 – valid
- 0x2 – suspended
- 0x4 – suspended by a standby-related action
- 0x8 – waiting for a marker
- 0x10 – will issue **dbcc ('ltm', 'ignore')**
- 0x20 – waiting for dump marker to initialize a standby database
- 0x40 – switching related duplicate detection when *ltype* is equal to 'P'
- 0x40 – allow switching when *ltype* is equal to 'L'
- 0x80 – temporarily not doing any grouping

Example:-

```
1> select * from rs_databases
2> go
dsname dbname dbid dist_status src_status attributes errorclassid funcclassid prsid
rowtype sorto_status ltype ptype ldbid enable_seq rs_errorclassid
-----
-----
AGSIT_DB_REP_ASA AGSIT_DB_REP_ASA 101 1 0 0 0x0000000001000002 0x0000000001000001
16777317 0 0 P A 101 0 0x000000000100001a
COMPOSER_DS SIT_Composer 102 17 17 0 0x0000000000000000 0x0000000000000000 16777317 1 0 L
L 102 0 0x0000000000000000
```

Checked for orphaned rows in rs_objects

```
select prsid, convert(char(30),objname), convert(char(30),phys_tablename), objid, dbid,
convert(char(30),deliver_as_name) from rs_objects
go
"16777317 rs_drp0x010000650000007a rs_drp0x010000650000007a 0x010000650000007a 102
rep_latency_tracking"
```

rs_drp0x0 is an internal repdef which belongs to 102. you can manually delete it, then, issue drop logical connection.