UniVerse Configuration
Blog Post 2
**Contents**

Introduction ................................................................................................................................. 3  
Disclaimer ................................................................................................................................... 3  
UniVerse 11.3 ............................................................................................................................... 4  
The UniVerse and UniData Memory Models ................................................................................. 5  
UniVerse and the New Shared Memory ......................................................................................... 6  
NUERS ....................................................................................................................................... 7  
SHM_GNPAGES and SHM_GNTBLS ......................................................................................... 9
Introduction

This is the second of two blogs on UniVerse configuration by Brian Leach for the Rocket MultiValue Blog. These posts concentrate on those configuration parameters that are related to performance and those that apply hard limits on the system.

In the first post, we looked at the configuration parameters introduced before UniVerse 11. Many of these have been part of UniVerse since its earliest times and reflect the limitations of earlier operating systems. Over time the defaults have changed to keep pace with new technology. At version 11, however, the UniVerse architecture has seen some fundamental changes and with those, the introduction of new configurable parameters of which administrators should be made aware.

Disclaimer

Please note that these parameters can fundamentally affect the way in which UniVerse and your applications will run, and should only be adjusted with caution. Configuration changes should be tested and Brian Leach Consulting Ltd. and Rocket Software Inc. will accept no responsibility for any damage or performance loss arising from your choice of parameters. If you are unsure of the changes you intend to make, seek professional advice.

Intended Audience

This series is pitched at experienced systems administrators and DBAs, and a working knowledge of UniVerse is assumed throughout.
When upgrading to UniVerse 11.x, you may have noticed the large number of changes. This has been one of the most substantial upgrades in the history of the product, and one that has introduced a number of foreign sounding concepts. You may have read the New Features guide and questioned about the changes in the LOGIN behaviour or the user numbering. You may have seen references to LCTs. In extreme cases, you may have been unlucky enough to have to deal with No More LCT or No More GCT errors. Welcome to the world of UniData.

Whilst outwardly similar, UniVerse and UniData do not share a common architecture. As a gross over-simplification you could say that historically, UniData has been built for volume and UniVerse for complexity. UniData places great store on being able to do very large numbers of simple things quickly, and optimises the available resources allocated to this task. UniVerse places equally great store on handling deep and meaningful tasks flexibly. Of course, this does not mean that UniData does not handle complexity, nor does it mean that UniVerse does not handle volume, but the impetus that drove the underlying architecture and the design decisions that informed it was different.

Until now, UniVerse DBAs have had the easier time. Other than the two hard stops in terms of configuration that we covered in the first post – the lock table and the T30FILE buffer – most UniVerse systems can run out of the box, albeit not optimally in terms of performance. UniData, however, imposes far more hard limits as the cost-side of its performance optimization, and UniData DBAs are accustomed to dealing with obscure tuning parameters.

With the introduction of U2 Replication, taken from UniData, into UniVerse 11; and future planning integration of other UniData features including the Recoverable File System to further strengthen the underlying data management, these parameters for the first time have been imported into UniVerse, along with accompanying useful features such as the cleanup deamon. UniVerse DBAs now need a broad understanding of how these come about, what new routines have been introduced as a result of the changes and how they are being applied to UniVerse.

So in this post, we will look into the new UniVerse configuration settings that derive from UniData. This will not be as detailed as they would be in a UniData manual, but should offer an insight and explain just what some of those cryptic messages mean.

Just to be clear, this is a fundamental architectural change and these new configurable parameters apply whether you are using U2 Replication or not.
The UniVerse and UniData Memory Models

Both UniVerse and UniData make use of two types of memory: shared and local (stack).

For UniVerse, the global shared memory segment has historically held the configuration details we covered in the previous post: login counts, semaphore tables, task locks, catalog shared routines, transaction logging buffers, printer information, port status information, NLS translation maps and so forth.

All this has been largely transparent to the UniVerse DBA except on the rare occasions when he or she has needed to remove the shared memory segment associated with an abnormally terminated process.¹

On UniData, there is a much greater emphasis on using shared memory pages, from everything from the lock and process management through to variable space in UniBasic programs. This prevents continuous reallocations and is, in theory, quicker: although UniData has similarly been revised at version 8 and now uses a more flexible UniVerse-like memory model for its Basic runtime.

¹ This should now be handled through the cleanup daemon.
UniVerse and the New Shared Memory

You may have noticed some new processes introduced in UniVerse 11. Chief amongst these are the uvsmm and uvcleanupd, which have direct counterparts in the UniData smm and cleanupd.

The Shared Memory Manager (uvsmm) is now responsible for creating the primary shared memory segment and managing the licensing, including several internal tables that track the various UniVerse processes.

If you don’t like Three Letter Acronyms, look away now.

On UniData, the smm daemon is responsible for allocating shared memory segments and pages (sub-divisions of segments used for processing) on demand. These are managed through a hierarchy of control tables, starting from the top-level CTL (Control Table List).

- The CTL contains a list of Global Control Tables (GCT).
- The GCT maps the shared memory segment into a set of global pages and records their use in Local Control Tables (LCTs). Global Pages are divided into local pages.
- The LCT stores details on each process group.
- The LCT holds the PI (Process Information)
- The LCT holds the MI (Memory Information)
- The LCT holds the CT (Counter Table)
- The LCT holds the CI (temporary allocation of memory pages)
- The LCT holds the dead process flag

The cleanup Daemon uses the dead process flag to safely clears up after any terminated processes so you should no longer need to explicitly remove shared memory segments and locks after an abnormal termination.
NUSERS

Now what, you might ask, has all this to do with UniVerse? Because of the architectural changes in 11.3, a lot of this is now reflected in UniVerse, most notably the LCT.

The LCT is a new hard limit on UniVerse systems. Prior to version 11, the only hard limits were independent of the number of concurrent process groups² (that is, the number of foreground users, API backend processes and phantom processes).

So long as they remain within the technology bounds of the iPhantom licencing (i.e. they do not open sockets, devices or other forms of connectivity that could masquerade as foreground users) you could fire off as many phantom processes as you wished to manage and offload batch, scheduled or long running operations without having to touch the configuration. Add in device licensing and even the foreground numbers can start to multiply up. UniVerse keeps track of these, but does not impose any limits that would cause those processes to fail so long as they are well behaved and do not fill up the lock table or exceed the runtime limits on concurrently open dynamic files.

From UniVerse 11, however, each of these must have a slot in the LCT to record their activity, and in common with all these shared memory arrays the LCT is created as a fixed size resource. The size of this array is given by the NUSERS configurable parameter. Here is what it says is the uvconfig file:

```
# The following parameters are borrow from UniData for UV Replication.
# NUSERS - Number of user sessions. It should be at least >= licensed users.
NUSERS   64
```

This is misleading. Yes, they are borrowed from UniData but not just for UV Replication. NUSERS is now a hard limit, though with a get-out clause: if the number given in the uvconfig file is less than (the number of licensed users * 3), it will be automatically increased on start-up to that higher number.

Quite where this multiplier was arrived at is anyone’s guess, but it is obviously intended to balance the number of users against the expected number of phantoms and device licensed entries and to give a fighting chance of not breaking large systems on upgrade.

² A Process Group refers to any top level process such as uv, uvsh or uvapi_slave; and the subordinate processes that it may launch as it performs its activities. When you run a verb like BASIC this spawns a separate process to run the nbasic executable then returns control back to the calling process once it has completed.
So what happens if you exceed that number? Any additional processes will fail to start and the following messages are recorded in the $UVHOME\uvsmm.errlog file:

Wed Oct 25 14:54:20 2017 - No more LCTs
Wed Oct 25 14:54:20 2017 - Please increase the uvconfig NUSERS setting, then stop UniVerse, run uvregen and restart UniVerse daemons or services.

You can see how close you are to the limit by using the $UVHOME\bin\uvlstt command:

C:\U2\UV>bin\uvlstt
----------------------- LCTs Statistics -----------------------
Total LCTs (Process Groups allowed): 64
LCTs Used (Active Process Groups): 2 (3% of 64)   Total Ps: 2
  Total Global Pages Used: 2 (256K bytes)
  Total Self-created.....: 0 (0K bytes)
  Total memory used......: 256K bytes
----------------------- End of LCTs Statistics -----------------------

The $UVHOME\bin\uvsm -l command shows the pid of the leading process in each of the LCT slots.

One other parameter that may potentially limit the number of processes is SHM_LPINENTS.

This holds the number of entries in the PI table inside each LCT, and therefore the maximum number of processes in each process group. Set to a default of 10 you would be very unlikely to hit this limit unless your application does a lot of shelling out.
SHM_GNPAGES and SHM_GNTBLS

Where the N USERS configurable parameter dictates the number of LCT segments and therefore the maximum number of sessions that UniVerse can run, the SHM_CNTBLS specifies the maximum number of GCT segments and therefore the maximum amount of shared memory that can be attached. Each shared memory segment is limited by SHM_MAX_SIZE.

To further complicate matters, shared memory segments are used on a page basis. Each segment is divided into pages that are attached and used on demand, using two more parameters: SHM_GNPAGES holds the number of pages in a segment and SHM_GPAGESZ the size of each page in blocks of 512 bytes.

It follows that \( \text{SHM_GPAGESZ} \times \text{SHM_GNPAGES} \times 512 \leq \text{SHM_MAX_SIZE} \).

The \($\text{UVHOME}\bin\text{uvgstt}$\) command shows details on global shared memory table use just following the default values added to the uvconfig file:

```
C:\U2\UV>bin\uvgstt
---------------------------- GCTs Statistics ---------------------------
Total GCTs (GSMs allowed): 20
Pages/GSM................: 32 (4096K bytes)
Bytes/Page..............: 128K bytes

GCTs used (GSMs created): 2 (10% of 20)
   Active GSMs....: 1 (32 pages in total, 4096K bytes)
       Pages Used........: 2 (6%, 256K bytes)
       Pages Freed.......: 30 (93%, 3840K bytes)
   Inactive GSMs..: 1
       Pages Freed.......: 32 (4096K bytes)

   Total Pages Used....: 2 (3%, 256K bytes)
   Total Pages Freed...: 62 (96%, 7936K bytes)
   Total memory allocated: 8192K bytes
---------------------------- End of GCTs Statistics ---------------------
```

As you can see from the details above SHM_MAX_SIZE is the maximum size limit and not the actual size that is reserved.

When do these global pages get used and do you need to care about them?
Administrators will have been accustomed to the fact that UniVerse processes have always required shared memory segment: every administrator will have seen the famous Printer Shared Memory Removed message when starting one UniVerse session over a badly terminated process.

These were historically created as separate segments identified by a segment key of 0xaceb followed by the port number in hexadecimal appearing in the `ipcs -m` listings, to be cleared up manually following a forced termination:

```
linux:/home/brian # kill -9 4058
linux:/home/brian # ipcs -m
------- Shared Memory Segments -------
key        shmid      owner      perms      bytes      nattch
0x00000000 360448     root      600        655360     2
0x00000000 393217     brian     600        262144     1
0xaceca200 819202     root      666        1008088    2
0xaceb0012 1048579    brian     600        10240      0
0xacebf023 1081348    brian     600        10240      1
```

```
linux:/home/brian # ipcrm -m 104958
```

This shared memory segment held coordination flags between processes in the same process group, such as the state of echoing, COMO, the DATA stack, terminal control structures and so forth, allowing UniVerse to fire off sub-processes without the need to use environment variables or other coordinating structures.

At UniVerse 11.3 these segments have disappeared from the `ipcs` listings. There are still the two central segments (0xacea and 0xacec) holding the global configuration and catalog shared memory respectively; but the individual segments have gone to be replaced by a more UniData-like arrangement of pages in a single large shared memory block. These can then be managed by the same `uvsmm` process and cleanup daemon.

Each UniVerse process now takes one shared memory page in a GSM (Global Shared Memory) segment: back to the acronyms I’m afraid. This is controlled by the configuration parameters `SHM_GNPAGES` and `SHM_GNTBLS` and their associated limits. These set out as follows:

**SHM_GNTBLS**

This holds the maximum number of GCT control entries and therefore the maximum number of global segments that can be attached. This should be a multiple of 32 and should not exceed the operating system limit (where applicable) of `shmmni` (maximum number of shared memory identifiers).
**SHM_MAX_SIZE**

This is the maximum shared memory size supported by the operating system.

**SHM_GNPAGES**

This holds the number of pages created in a segment and should also be a multiple of 32.

**SHM_GNPAGESZ**

This holds the size of each global page in 512 byte blocks. It follows that \( \text{SHM_GPAGESZ} \times \text{SHM_GNPAGES} \leq \text{maximum shared segment size} \).

That’s all probably as clear as mud. The key indicator that you need to look at these is the appearance of a typically helpful UniData style message: No More GCTs. This means you have run out of available global memory pages to host the per-process sessions.

SHM_GNPAGESZ should not need tuning as it has already been set to the correct size for a UniVerse segment (256). In cases where you are hitting up against this error, you will need to tune SHM_GNPAGES and SHM_GNTBLS to ensure you have sufficient number of pages available.

Rocket have managed this by increasing the defaults to handle around 97% of customer reported GCT and LCT errors on UniVerse, and the other shared memory parameters - most of which had to do with memory allocation for the Basic run machine on UniData - are irrelevant and can be kept at their default values. The defaults are now 2560 GCTs, made up of SHM_GNPAGES of 64 * SHM_GNTBLS of 40. Remember as always that this includes all sessions including UniObjects connections, ODBC connections, phantoms and the like.