Enabling Service Excellence

Monitoring Rocket UniVerse Database Server

eG Enterprise v6

Table of Contents

1

Monitoring the UniVerse Database Server

UniVerse is a database, and also a platform for powerful and feature rich business applications. UniVerse is a MultiValue Database Management System (MVDBMS). MultiValued data structures support nested entities removing the need to normalize information. These provide storage of complex data types, and allow for a more natural representation of real world entities such as sales orders or manufacture assemblies. UniVerse also features an embedded business language suitable for creating complete applications or for encapsulating complex business rules in client/server or internet based solutions, and in service orientated architecture (SOA). Because of these unique features, UniVerse databases are frequently found as embedded data sources behind mission-critical business, financial, engineering and government systems. If such a database is unavailable over the network, even for a brief while, or is not sized/configured right to handle the session load, or experiences frequent locks, it can adversely impact the uptime of and user access to the mission-critical services it supports. If such an undesirable outcome is to be avoided, the UniVerse database server needs to be continuously monitored.

eG Enterprise provides a specialized monitoring model for the UniVerse Database server. Each layer of this model is mapped to tests that employ native operating system-level commands, network-level pings and UniVerse shell commands to periodically check on the availability of the database server, track the session load, and look for file locks in the database. Whenever an abnormality is detected – say, a session overload condition on the database, a sudden break in the availability of the server, or many and frequent file locks – the tests promptly notify the database administrators, so that they can instantly initiate remedial measures.

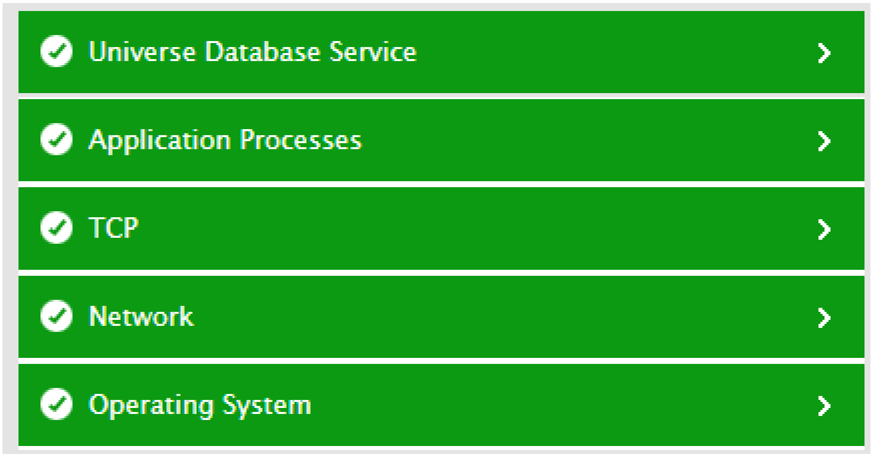

Figure 1.1: Layer model of the UniVerse Database server

The sections that follow will discuss the first layer of Figure 1.1 only, as all other layers have already been dealt with in the *Monitoring Unix and Windows Servers* document.

1.1 Universe Database Service Layer

Using the tests mapped to this layer, the session load on the server and the impact of the locking activity on the sessions can be ascertained.

1.1.1 UniVerse Database Sessions Test

Administrators can limit the number of sessions that can be established on a UniVerse database server, so as not to choke the server. At frequent intervals, administrators should monitor the session count on the server and figure out if the maximum session limit is about to be reached or not. This will enable administrators to proactively detect potential overload conditions, and take pre-emptive action against the same. This is exactly what the **UniVerse Database Sessions** test helps administrators achieve! At configured intervals, this test reports the session load and overall session usage by the users of the database server, and warns administrators of a probable overload condition on the server. Additionally, the test also reports the number and type of sessions launched per user, and thus reveals which user(s) has contributed the most to the overload and the type of sessions responsible for the same. The details of users and their sessions are also revealed as part of detailed diagnostics. Using these useful problem pointers, administrators can decide between killing idle sessions to reduce the load on the server or increasing the session limit to accommodate more number of sessions.

Target of the Test: A UniVerse database server

Agent running the test: An internal agent

Output of the test: One set of results for every user who is currently logged into the database server. Measures will also be reported for an additional *All* descriptor. The session load and usage metrics will be aggregated across all users and reported for this descriptor.

Parameters of the test:

1. **TEST PERIOD** - How often should the test be executed
2. **HOST -** The host for which the test is to be configured.
3. **PORT** – The port at which the **HOST** listens. By default, this is *31438*
4. **UNIVERSE SHELL PATH** – This test uses UniVerse shell commands to pull the desired metrics from the server. To enable the test to run these commands, provide the path to the *bin* folder of the UniVerse install directory. For example, for a Windows installation of UniVerse, the **UNIVERSE SHELL PATH** can be: *C:\U2\UV*. For a Unix installation, your specification can be: */usr/uv*
5. **ECLIPSE PATH** - U2 DBTools include Eclipse-based tools for programming and administration. Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications. Where Eclipse IDE is installed, the eG agent should be configured to use the Eclipse environment for executing the UV shell commands. In this case therefore, type the full path to the Eclipse home directory. By default however, this parameter is set to *none*, indicating that by default, the eG agent runs the commands from the UniVerse shell only and not Eclipse.
6. **INCLUDE LOGGED IN TIME** – This test reports detailed diagnostics for the *Interactive sessions* and *Background sessions* measures. By default, the details of these sessions – eg., the user who launched the sessions, the commands last executed in the sessions, and the terminal that the user logged in from, will be reported as part of detailed diagnosis. The login time of the user will however, not be reported as part of the detailed measures by default. This is why, the **INCLUDE LOGGED IN TIME** flag is set to **No** by default. To make sure that the detailed diagnosis includes the login time of the user as well, set this flag to **Yes**.
7. **DD FREQUENCY** – Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency.
8. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Metrics reported by the test:

Measurement	Measurement Unit	Interpretation
Maximum session limit: Indicates the maximum number of sessions that can be launched on the target UniVerse database server.	Number	**This measure will only be reported for the 'All' descriptor of this test.**
Total sessions: Indicates the total number of sessions currently opened by this user. For the *All* descriptor, this measure will report the total number of open sessions on the target UniVerse database server, regardless of user.	Number	Using the value that this measure reports for the *All* descriptor, you can gauge how loaded the server is presently. You can then compare the value of this measure with the value of the *Maximum session limit* measure to know if the server is fast approaching its session limit or can accommodate many more sessions. If the former is true, it indicates a potential overload condition on the server. You can then compare the value of this measure across users to know which user is contributing the most to the overload.
Session usage: Indicates the percentage of sessions utilized by this user. For the *All* descriptor, this measure will report what percentage of the maximum session limit is being used by all users logged into the server.	Percent	A value close to 100% for the *All* descriptor reveals that too many sessions are currently open on the server, and may soon cause the server to consume its session limit. If that happens, then the server will deny access to new session requests. To avoid this, first check whether the server is sized with adequate resources to handle additional load, and if so, increase the session limit of the server. However, if the server does not have enough processing power, then you may want to kill the sessions that are idle or are engaged in inconsequential operations, so that the load on the server drops; this way, you can make room for newer sessions as well. To achieve this, first compare the value of this measure across users to know which user is contributing to the overload. Then, use the detailed diagnosis of the *Interactive sessions* and *Background processes* measures of that user to know which sessions of these users are unnecessarily consuming resources, and target such sessions for termination.

	Interactive sessions: Indicates the total number of interactive processes initiated by this user. For the *All* descriptor, this measure will report the sum total of interactive processes initiated by all users of the database server.	Number	Interactive processes are those that are initiated by the user and run in the foreground. Background processes are those that are initiated by the user using the PHANTOM command. Phantom processes run in the background and are useful for long running or load spreading operations. If the value of the *Session usage* measure is close to 100% for the *All* descriptor, then compare the value of the *Interactive sessions* and *Background sessions* measure for that descriptor to know what type of sessions are causing the overload. If interactive sessions are contributing to the overload, then compare the value of the *Interactive sessions* measure across users to know which user has initiated the maximum number of interactive processes. Then, use the detailed diagnosis of the *Interactive sessions* measure of that user to view the PID of the interactive processes, the command that was last executed by each process, which terminal that user logged in from, and what time the user logged in. From this, you can identify the interactive sessions that have been open for too long a time running inconsequential commands, and mark them for termination. Likewise, if background sessions are contributing to the overload, then compare the value of the *Background sessions* measure across users to know which user has initiated the maximum number of background processes. Then, use the detailed diagnosis of the *Background sessions* measure of that user to view the PID of the background processes, the command that was last executed by each process, which terminal that user logged in from, and what time the user logged in. From this, you can identify the background sessions that have been open for too long a time running inconsequential commands, and mark them for termination.
	Background sessions: Indicates the number of background processes initiated by this user. For the *All* descriptor, this measure will report the sum total of all background processes initiated by all users of the database server.	Number	

1.1.2 UniVerse Database Active Group Locks Test

UniVerse stores its data in containers known as Files. Static and Dynamic files are the most common files in a UniVerse database. Both static and dynamic files consist of a number of separate storage sections, known as buffers. The size and number of these buffers is specified when the file is created, and space is reserved for these on disk. This is known as the 'primary space' of the file. As records are added to the file, UniVerse distributes the records across this primary space using a scattering formula based on the record keys. This formula produces the address of the buffer in which the record should be placed. If a buffer in a file becomes full, and another record is added to the file that is addressed to the same buffer, the file is said to be 'overflowed'. It cannot use a different buffer to hold the record, since this would break the allocation scheme. Instead it will extend the file by chaining an additional buffer to the end of the first buffer – and so on, so that more records are added to the file. The chain of buffers attached to a specific buffer in the primary space, is known as a 'group'.

When a process needs to read or write a record to or from a file, UniVerse first works out which group should hold the record to be written or holds the record to be read in the primary space by applying the relevant hashing algorithm to the record key. Because the group structure can hold more than one record, UniVerse needs to ensure that only one process at a time can read from, or update an individual group. This prevents one process trying to scan through a group whilst another process is busy reorganizing it. So it then proceeds to place a lock entry for that group into the lock table. Because the lock specifies both the file and the group within the file, this does not prevent another UniVerse process from accessing a different group in the same file at the same time. So different processes can still read and write a single file safely.

If a file is well sized and records are distributed across the primary space, it should be very rare that two processes will need access to the same part of the file at the same time, and so there will be few collisions within the group lock table. If a file is badly sized and records are consigned to long overflow chains of buffers belonging to the same group, the chances of more than one process needing access to the same group are much higher – and so the collisions are much higher. And the longer the chains, the longer the time that each lock needs to be held. All of which means that more processes will spend longer queuing up and waiting for the lock to become available.

This is why, if a processing slowdown is noticed in a UniVerse database, administrators should first check whether too many group locks are currently active in the database, and identify the mode of these locks. This is where the **UniVerse Database Active Group Locks** test helps. For each lock mode, this test reveals the number of active group locks in that mode. The detailed diagnostics provided by this test provide deep insights into these locks, and in the process, enables administrators to figure out:

- Which processes are holding what type of lock?
- Which process is holding a lock on which group of which file?

With the help of these details, administrators can identify processes that are unnecessarily holding a lock, and can unlock the files and groups so locked, so as to ease processing.

Target of the Test: A UniVerse database server

Agent running the test: An internal agent

Output of the test: One set of results for every group lock mode currently active in the target database. The table below discusses the probable locking modes and what they represent:

Descriptor	Description
EX	Exclusive update lock
SH	Shared lock
RD	Read lock
WR	Write lock
IN	Information lock

Parameters of the test:

1. **TEST PERIOD** - How often should the test be executed
2. **HOST -** The host for which the test is to be configured.
3. **PORT** – The port at which the **HOST** listens. By default, this is *31438*
4. **UNIVERSE SHELL PATH** – This test uses UniVerse shell commands to pull the desired metrics from the server. To enable the test to run these commands, provide the path to the *bin* folder of the UniVerse install directory. For example, for a Windows installation of UniVerse, the **UNIVERSE SHELL PATH** can be: *C:|U2|UV*. For a Unix installation, your specification can be: *|usr|uv*
5. **ECLIPSE PATH** - U2 DBTools include Eclipse-based tools for programming and administration. Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications. Where Eclipse IDE is installed, the eG agent should be configured to use the Eclipse environment for executing the UV shell commands. In this case therefore, type the full path to the Eclipse home directory. By default however, this parameter is set to *none*, indicating that by default, the eG agent runs the commands from the UniVerse shell only and not Eclipse.

6. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Metrics reported by the test:

Measurement	Measurement Unit	Interpretation
Active group locks: Indicates the number of active group locks in this mode.	Number	Compare the value of this measure across modes to identify the mode which most of the group locks are in. If the database server has been frequently experiencing processing bottlenecks, you can use the detailed diagnosis of this measure to identify the processes that are holding the locks in that mode, the users who initiated the processes, the files and groups that are locked, the file system in which the files exist, and the host from which the lock originated. With the help of this information, administrators can instantly isolate the processes that are holding the locks unnecessarily and the users who are running those processes. Such processes can later be killed or the lock they hold released to clear the processing bottleneck.

1.1.3 UniVerse Database Active Record Locks Test

UniVerse record and file locks control access to records and files among concurrent user processes. To control access to records and files, UniVerse supports two levels of lock granularity:

- Fine granularity of record locks
- Coarse granularity of file locks

Granularity refers to the level at which a process or program acquires a lock. Record locks affect a smaller element, the record, and provide a fine level of granularity, whereas file locks affect a larger element, the file, and produce a coarse level of granularity. Lock compatibility determines what a user's process can access while other processes have locks on records or files. Record locks allow more compatibility because they coexist with other record locks, thus allowing more transactions to take place concurrently. However these "finer-grained" locks provide a lower isolation level. File locks enforce a high isolation level, more concurrency control, but less compatibility. Lock compatibility decreases and isolation level increases as strength and granularity increase. This may increase the possibility of deadlocks at high isolation levels. Within each granularity level, the strength of the lock can vary. UniVerse supports the following locks (in order of increasing strength):

- Shared record lock
- Update record lock
- Shared file lock
- Intent file lock
- Exclusive file lock

The locks become less compatible as the granularity, strength, and number of locks increase. Therefore the number of lock conflicts increase, and fewer users can access records and files concurrently. Weaker locks can always be *promoted* to stronger locks or *escalated* to a coarser level of granularity if needed.

Whenever users complaints regarding data inaccessibility increase, it is good practice for administrators to check if too many file/record locks are being held, and if so, what their strength is. The **UniVerse Database Active Record Locks** test provides administrators with this information, instantly! This test automatically discovers the types of locks currently active in the target UniVerse database and reports the count of locks held per type. From the lock types, administrators can quickly infer the strength of the locks. If too many strong locks are active on the database, then the detailed diagnostics provided by this test can be used to identify the processes holding the locks, the users who own the processes, and the file and records locked. This way, administrators can precisely isolate those processes that are unnecessarily holding the locks and can initiate measures to have those locks released.

Target of the Test: A UniVerse database server

Agent running the test: An internal agent

Output of the test: One set of results for every lock type currently active in the target database. The table below discusses the probable lock types and what they represent:

Descriptor	**Description**
RU	Update record lock
RL	Shared record lock
FS	Shared file lock
IX	Shared file lock with intent to acquire an exclusive file lock
FX	Exclusive file lock
XU	Exclusive lock set by CLEAR.FILE
CR	Shared file lock set by RESIZE
XR	Exclusive file lock set by RESIZE

Parameters of the test:

1. **TEST PERIOD** - How often should the test be executed
2. **HOST -** The host for which the test is to be configured.
3. **PORT** – The port at which the **HOST** listens. By default, this is *31438*
4. **UNIVERSE SHELL PATH** – This test uses UniVerse shell commands to pull the desired metrics from the server. To enable the test to run these commands, provide the path to the *bin* folder of the UniVerse install directory. For example, for a Windows installation of UniVerse, the **UNIVERSE SHELL PATH** can be: *C:|U2|UV*. For a Unix installation, your specification can be: *|usr|uv*
5. **ECLIPSE PATH** - U2 DBTools include Eclipse-based tools for programming and administration. Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications. Where Eclipse IDE is installed, the eG agent should be configured to use the Eclipse environment for executing the UV shell commands. In this case therefore, type the full path to the Eclipse home directory. By default however, this parameter is set to *none*, indicating that by default, the eG agent runs the commands from the UniVerse shell only and not Eclipse.

6. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Metrics reported by the test:

Measurement	Measurement Unit	Interpretation
Active record locks: Indicates the number of active file/record locks of this type.	Number	Compare the value of this measure across types to determine the strength of the majority of locks currently held. The table below discusses how each type of locks impact user access:

Lock type (strength)	**Allows other users to acquire**	**Prevents other users from acquiring**	**Is ignored if the current user already owns**
Shared file lock	• Shared record lock • Shared file lock • Intent file lock	• Update record lock • Exclusive file lock	• Shared record lock • Update record lock • Shared file lock • Intent file lock • Exclusive file lock
		•	•

Lock type (strength)	Allows other users to acquire	Prevents other users from acquiring	Is ignored if the current user already owns
Update record lock	No locks	• Shared record lock • Update record lock • Shared file lock • Intent file lock • Exclusive file lock	• Update record lock • Exclusive file lock
Shared file lock	• Shared record lock • Shared file lock	• Update record lock • Intent file lock • Exclusive file lock	• Shared file lock • Intent file lock • Exclusive file lock
Intent file lock	Shared record lock	• Update record lock • Shared file lock • Intent file lock • Exclusive file lock	• Intent file lock • Exclusive file lock

Lock type (strength)	**Allows other users to acquire**	**Prevents other users from acquiring**	**Is ignored if the current user already owns**
Exclusive file lock	No locks	• Shared record lock • Update record lock • Shared file lock • Intent file lock • Exclusive file lock	• Exclusive file lock

If too many strong locks are held, you can use the detailed diagnosis of this measure to identify the processes that are holding the strong locks, the users who initiated the processes, the files and records that are locked, the file system in which the files exist, and the host from which the lock originated. With the help of this information, administrators can instantly isolate the processes that are holding the locks unnecessarily and the users who are running those processes. Such processes can later be killed or the lock they hold released to clear processing bottlenecks (if any).

Conclusion

This document has described in detail the monitoring paradigm used and the measurement capabilities of the eG Enterprise suite of products with respect to the **UniVerse Database Server**. For details of how to administer and use the eG Enterprise suite of products, refer to the user manuals.

We will be adding new measurement capabilities into the future versions of the eG Enterprise suite. If you can identify new capabilities that you would like us to incorporate in the eG Enterprise suite of products, please contact support@eginnovations.com. We look forward to your support and cooperation. Any feedback regarding this manual or any other aspects of the eG Enterprise suite can be forwarded to feedback@eginnovations.com.